# Fault Diagnosis of Combinational Networks - A Study of Graph-Theory Methods

A THESIS SUBMITTED
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

**MASTER OF TECHNOLOGY**
IN
ELECTRICAL ENGINEERING

BY
**SYED ASHRAF ALI**

EE
1973
M
ALI
FAU

TO THE

DEPARTMENT OF ELECTRICAL ENGINEERING
**INDIAN INSTITUTE OF TECHNOLOGY, KANPUR**
**September 1973**

# Fault Diagnosis of Combinational Networks - A Study of Graph-Theory Methods

A THESIS SUBMITTED
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

**MASTER OF TECHNOLOGY**
**IN**
ELECTRICAL ENGINEERING


**BY**
**SYED ASHRAF ALI**


TO THE


DEPARTMENT OF ELECTRICAL ENGINEERING
**INDIAN INSTITUTE OF TECHNOLOGY, KANPUR**


**September 1973**

I.I.T. KANPUR
CENTRAL LIBRARY
Acc. No. A28384

11 MAR 1974

V
JUNE '76

V
'76

Submitted 13.9.73

## CERTIFICATE

This is to certify that the thesis entitled "Fault Diagnosis of Combinational Networks - A Study of Graph-theory Methods" is a record of the work carried out by Mr. S.A. Ali under my supervision and that it has not been submitted elsewhere for the award of a degree.

Narsingh Deo

Dr. Narsingh Deo
Associate Professor
Department of Electrical Engg. and
Computer Centre
Indian Institute of Technology
Kanpur

Kanpur
1st Sept.1973

POST GRADUATE OFFICE
This thesis has been approved for the award of the Degree of Master of Technology (M.Tech.) in accordance with the regulations of the Indian Institute of Technology Kanpur
Dated. 1.1.74

## ACKNOWLEDGEMENTS

Without the encouragement and advice, of Prof. Narsingh Deo, this work would not have been possible. I am grateful to him for the patience with which he discussed each problem.

Thanks are due to my friends, especially Mr. M.S. Krishnamoorthi, who helped in the preparation of the thesis.

Finally, Mr. S.V. Sirbhai, must be thanked for the excellent typing.

TABLE OF CONTENTS


| | | Page |
|---|---|---|
| CHAPTER 1 | INTRODUCTION | 1 |
| 1.1 | Plan of the thesis | 2 |
| 1.2 | Definitions and Assumptions | 2 |
| CHAPTER 2 | FAULT DIAGNOSIS OF COMBINATIONAL NETWORKS | 6 |
| 2.1 | Various methods | 6 |
| CHAPTER 3 | GRAPH-THEORY IN FAULT DIAGNOSIS | 20 |
| 3.1 | Structural Models | 21 |
| 3.2 | Behavioural Models | 41 |
| CHAPTER 4 | TEST GENERATION FROM THE DIGRAPH MODEL OF A NETWORK | 48 |
| 4.1 | Conversion of logic description into a digraph | 48 |
| 4.2 | Path generating method | 51 |
| 4.3 | Example | 54 |
| CONCLUSION | | 63 |
| REFERENCES | | 64 |

# SYNOPSIS

Combinational networks constitute an important part of any digital computer. Therefore, they must be frequently tested for fault-free operation. The testing scheme must include a set of tests which will detect and locate the faulty unit. This thesis is an attempt to study the graph-theory methods for fault diagnosis of combinational networks. The advantages which the graph-theory methods offer are also studied. Some changes in the network design have to be incorporated in order to obtain reliable and fast fault location. These changes are determined by studying the structural properties of the network using a graph-theoretic model.

A method to obtain the digraph model of the network from its wiring description is presented. Its use in obtaining a minimal set of tests using the method of Koga et.al. is illustrated. It is found that, for large networks having few redundancies and reconvergent paths, the method of Koga et.al. provides the most efficient means to generate a minimal set of tests.

# CHAPTER 1

## INTRODUCTION:

A major requirement of a digital computer is the ability to operate correctly over sufficiently long periods of time. In order to fulfil this requirement the computer is tested frequently. On the detection of a fault corrective action is taken. This corrective action includes location of the faulty unit for repair or replacement.

Combinational networks constitute an important part of any digital computer. Therefore, along with the other parts of the computer, these also should be tested. The testing is carried out at the different stages, namely, production and field maintenance. The testing scheme at any stage must include a set of tests which will detect and locate the faulty unit. This thesis is an attempt to study the graph-theory methods for fault diagnosis of combinational networks. The advantages which the graph-theory approach offers are also studied. Some changes in the network design may be incorporated in order to obtain reliable and fast fault-location. These changes are determined by studying the graph-theoretic model of the network.

## 1.1 Plan of the Thesis:

In order to discuss the advantages of the graph-theory approach over the presently used methods, a review of these methods is made in Chapter 2.

Graph-theory methods are described in Chapter 3.

In Chapter 4, a modified form of the path-generating method (introduced in Chapter 3) is described. This method generates a minimal set of tests directly from the wiring description of the network.

## 1.2 Definitions and assumptions:

Fault: A fault in a combinational network is a condition which causes an output of the network to depart from its normal value.

The type of faults that we shall be concerned with are logical faults, that is, faults which produce a change in the logical behaviour of the network.

The class of faults to be considered in testing a combinational network depends on the type of logic circuits which constitute the network. However, most of the faults in currently used circuits like diode-resistor logic (DRL) circuits, diode-transistor logic (DTL) circuits, transistor-resistor logic (TRL) circuits and transistor-transistor logic (TTL) circuits, can be represented by an input or output of some gate being stuck-at-one (s-a-1) or stuck-at-zero(s-a-o).

That is, an input or output of some gate may assume a fixed logical value, independent of the inputs applied to the network. This covers most of the faults in networks - with discrete components as well as integrated circuits.

The malfunctioning of a logic network may be due to a permanent fault or an intermittent one. Although intermittent faults do occur in practice, the theory is lacking in procedures for testing them. Since the fault may disappear when a test is applied, there are no reliable methods of detecting their occurrence. Therefore, here we shall be concerned only with permanent faults, which do not disappear or change their nature during testing.

Only a single-fault is assumed to be present at any time in the network. This assumption is justified because of the frequent testing during field maintenance. However, during production multiple-faults are bound to occur because of errors in wiring, etc. The tests generated using the single-fault assumption can be applied in this stage by the start-small approach. In this method the smallest possible portion of the network is tested first. Each additional test adds a small increment to the circuitry tested. When a given test detects a fault, the assumption is that the failing circuit is within the group of circuits added by that test. The testing is sequential, in that, whenever a fault is detected

the faulty circuit is repaired and then only further testing is done. This approach is therefore suitable for multiple-fault detection and correction.

Test: By definition, some of the outputs of a faulty network will be different from their normal values for some input combination. Hence, any such input combination is a test which will detect the presence of a fault.

Requirements of tests:

To be effective the set of tests must be complete, that is, it should detect any fault which is likely to occur and also be able to locate the faulty unit. The set of tests should also be as small as possible, in order to minimize the time required for checking out.

The presence of a redundant element in the network inhibits the testing procedure. Because, the redundant element masks the failure of a non-redundant element. Hence, the fault cannot be detected and no corrective action can be taken.

Any irredundant combinational binary network with n input lines can be completely tested by applying all $2^n$ input combinations to it. This method is simple and can be readily implemented. But, it is too lengthy. Allowing a test rate of $10^6$ tests/second and fifty inputs (n=50), the total time it takes to test exhaustively with all possible combinations,

is around 50 years. A smaller set of tests is often sufficient for detecting all faults that are likely to occur. Thus one of the important problems associated with the testing of large combinational networks is that of generating a minimal set of fault-detection tests.

In addition to detecting faults in a network we must locate the fault to a specific gate or a replaceable module. Here also an efficient set of fault-location tests are required.

In the subsequent chapters we deal with the methods available for generating fault-detection and fault-location tests.

# CHAPTER 2

## FAULT DIAGNOSIS OF COMBINATIONAL NETWORKS

### 2.1 Various Methods:

A number of methods to detect and to locate faults in combinational networks are available [2]. These are:

(1) Partitioning or truth-table method

(2) Fault-matrix method

(3) One-dimensional path sensitizing method

(4) d-Algorithm method

(5) Boolean-difference method

(6) Graph-theory methods

Presently the most commonly used methods are 2-5.

(1) Partitioning or Truth-Table method

The method consists of simulating the behaviour of the normal network along with that of N previously defined faulty versions, by their truth-tables. The response of each of the versions to a set of input combinations is used to affect a partitioning of the (N+1) network versions, such that:

(i) The fault-free network is distinguished from all others, that is, the fault is detected.

(ii) All the (N+1) versions are uniquely identified, that is, the fault is detected and located.

The resulting partitioning defines a set of tests which will detect and locate the fault. Such a set of tests

is termed as a Diagnostic Test Sequence (DTS).

(2) Fault-Matrix method [4,9]

This is an extension of the truth-table method. It provides means for finding a minimum set of tests, for fault detection and location, from the $2^n$ possible input combinations.

The fault-matrix F relates a set of tests to their associated faults. For any entry F(i,j) of the fault-matrix,

$F(i,j) = 1$ if the fault $f_j$ is detected by the test $t_i$ and, $F(i,j)=0$ otherwise.

The fault-matrix is obtained from the truth-tables of the fault-free network and the N possible faulty versions of the network. For a network having m outputs, the combined truth-table has $2^n$ rows and (N+1). m columns, in sets of m columns. The first set of m columns represents the truth-table of the fault-free network. The procedure to obtain the fault-matrix from the combined truth-table, follows:

(i) Take the bitwise exclusive-OR of the first set of m columns with each of the other N sets.

(ii) OR the results of step 1 on a bitwise basis.

(iii) The resulting matrix is the fault-matrix and it has $2^n$ rows and N columns.

Identical columns in the fault-matrix indicate that the corresponding faults are indistinguishable. Each set

of identical columns is replaced by a single column. A column of all zeros indicates a fault which cannot be detected by any test. Each set of identical rows is replaced by a single row. A row of all zeros or all ones indicates a test which cannot distinguish between any two faults. Hence, it is removed. A reduced fault-matrix is thus obtained. The procedure to obtain a minimal DTS from this matrix is described below.

(i) Choose the test $t_i$ which distinguishes the maximum number of pairs of faults [4]. The chosen test $t_i$ must have the maximum weight, where the weight $W_i$ of the test $t_i$ is defined as in equation 2.1.

$$W_i = w_i(o) \cdot w_i(1) \qquad (2.1)$$

where $w_i(o)$ = Number of faults which are not detected by test $t_i$

$w_i(1)$ = Number of faults which are detected by test $t_i$

The criteria of choosing the test with minimum $|w_i(o) - w_i(1)|$, given in [9], gives the same results as equation 2.1.

(ii) Application of the chosen test $t_i$ will partition the fault-matrix into two. One submatrix corresponding to the faults which are detected by test $t_i$. The other submatrix corresponding to the faults which are not detected by test $t_i$.

(iii) From each submatrix of the fault-matrix a test having the maximum weight is chosen, and further partitions are made. This process is continued until each submatrix corresponds to a single-fault or a set of indistinguishable faults.

The DTS is represented as a rooted tree. The nodes of the tree represent the tests applied and the edges represent the outcome of each test, that is, a fault is detected or not. This tree representation of the DTS is referred as a diagnosing tree. The number of levels in the diagnosing tree is a measure of the time of application of the DTS.

Applying the procedure described, to the fault-matrix in table 2.1, a minimal DTS obtained is $(t_2, t_3, t_4, t_5, t_7)$. The order in which the tests are applied is represented as a diagnosing tree in figure 2.1.

F=

| | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ | $f_6$ | $f_7$ | $f_8$ | $f_9$ | $f_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $t_1$ | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| $t_2$ | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| $t_3$ | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| $t_4$ | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $t_5$ | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| $t_6$ | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| $t_7$ | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| $t_8$ | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

Table 2.1 Example fault-matrix

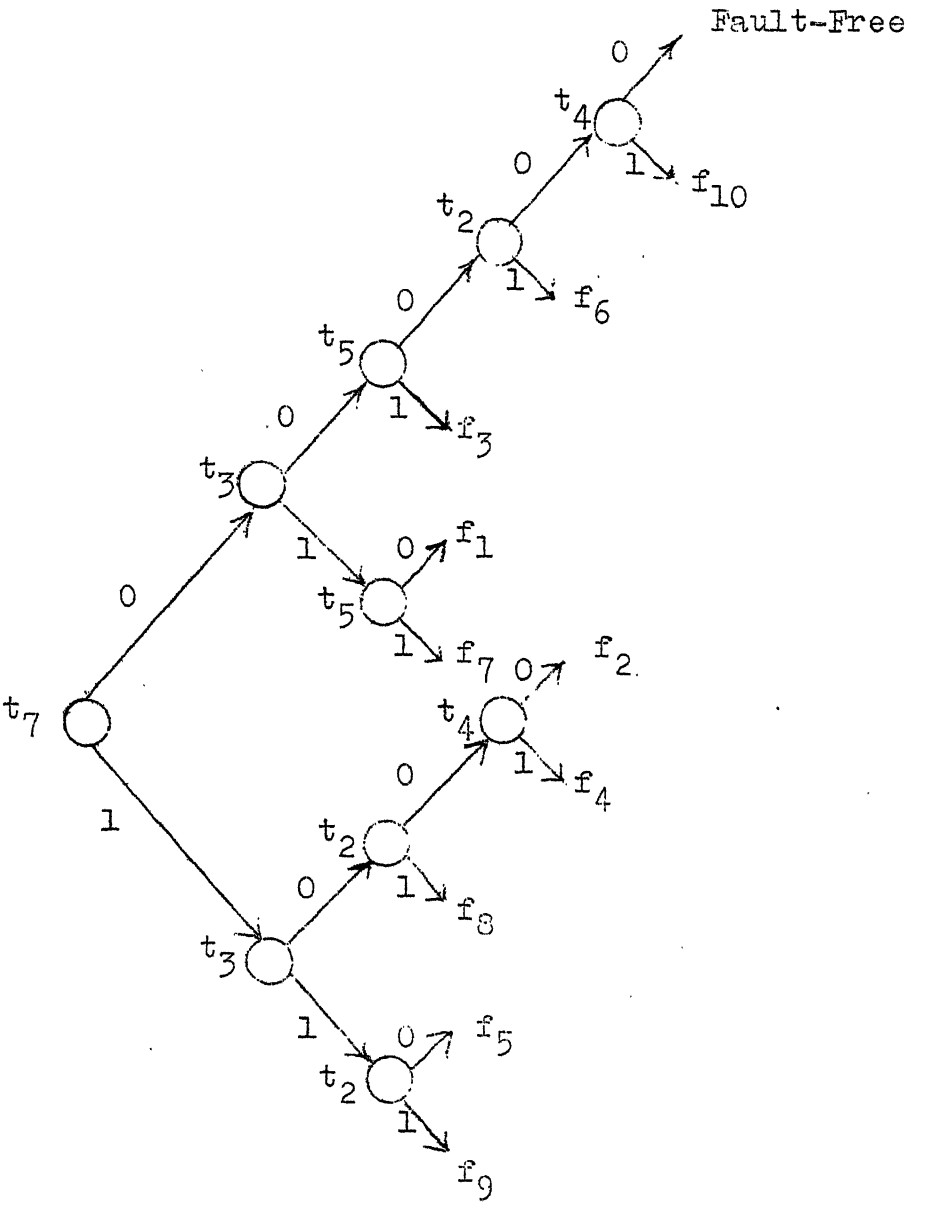



Figure 2.1 Diagnosing tree representation of the minimal DTS

To obtain the fault-matrix, an exhaustive enumeration of the network response, to all $2^n$ possible tests, is required for each fault. Only after this can the minimizing process, to obtain a minimal DTS, be carried out. Therefore, this method though simple and effective is not suitable for networks having a large number of inputs ($n > 10$).

(3) One-dimensional Path Sensitizing:

A sensitized path from the site of a fault to one of the network outputs, is one along which any change in the logical value at the site of the fault will result in a corresponding change at the output. The one-dimensional path sensitizing method for generating a DTS is based on three processes.

(i) The postulation of a known fault at a known site.

(ii) The propagation of the fault from its site to one of the network outputs via a sensitized path. This is termed as the forward trace phase.

(iii) Assigning values to the network inputs that yield the required sensitized path. This is termed as the backward trace phase.

A disadvantage of this method is that it may fail to detect a fault, even in the presence of sensitized paths to one of the outputs. This failure is caused by the reconvergent fan-out paths, with unequal parity of inversions, present in the network. Also, a single sensitized path may not be able to

detect the fault, while the fault can be detected by sensitizing more than one path simultaneously [17].

An example from [5] illustrates the one-dimensional path sensitizing method and its limitations.

Consider the network shown in figure 2.2. We want to generate a test to detect the fault of output of gate $E_2$ s-a-1.

(i) Set the output of $E_2$ opposite to the fault, that is,

$$E_2 = 0 \quad \text{or} \quad x_3 = 1.$$

(ii) Arbitrarily sensitize the path $E_2\ E_5\ E_6\ E_8$ to the output $Z_2$. For this, the output of each gate in the path should be dependent only upon the input coming from the previous gate in the sensitized path.

Set, $x_4 = 1$ to sensitize path through $E_5$,
and $x_5 = 0$ to sensitize path through $E_6$,
also, $E_3 = 1$ (output of gate $E_3$) to sensitize path through $E_8$. This completes the forward trace phase.

(iii) In the backward trace phase, we have to assign values to the network inputs such that $E_3 = 1$. For the given example, $x_1 = 0$ and $x_2 = 1$ is one possibility. Hence, the input combination (01110) is a test to detect the fault, output of $E_2$ s-a-1. This test will also detect the faults, output of $E_5$ s-a-1, $E_6$ s-a-1 and $E_8$ s-a-1.

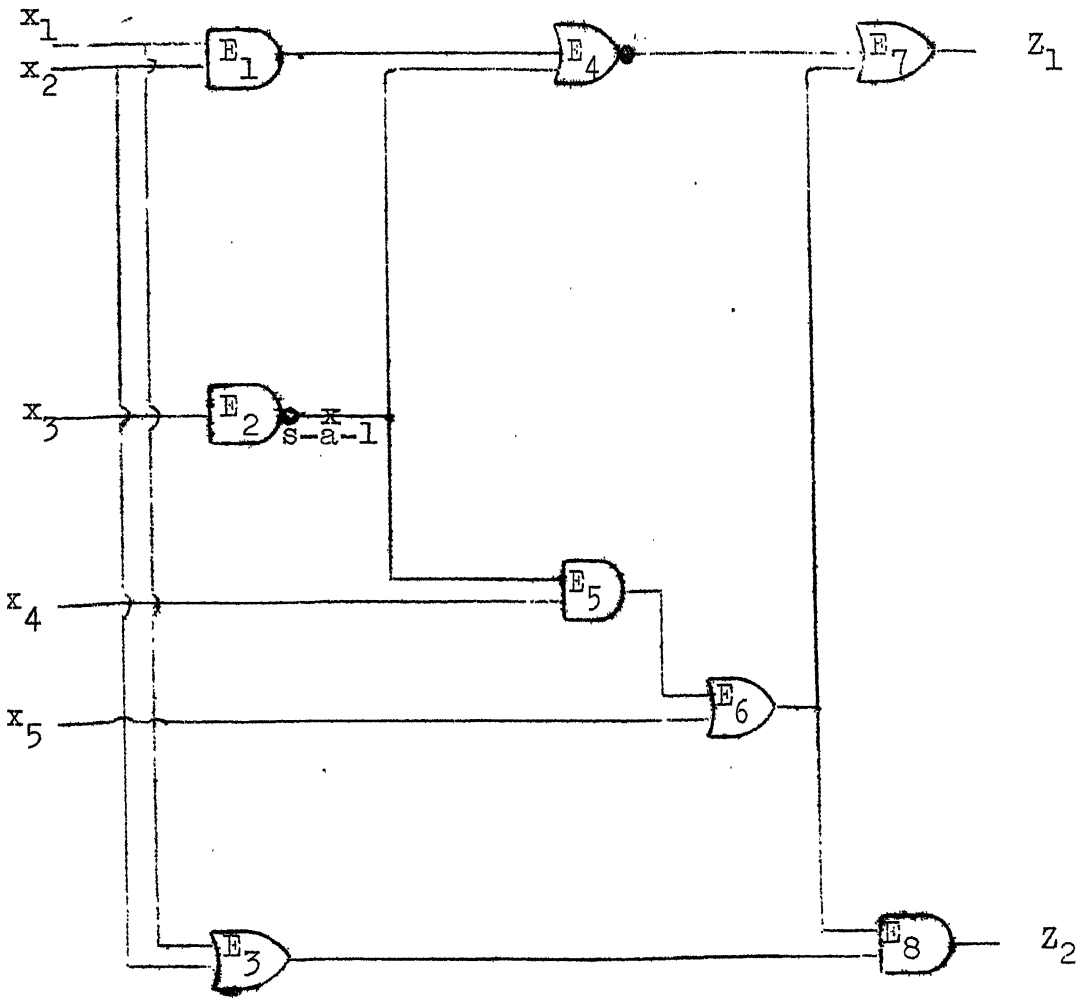



Figure 2.2 Example network to illustrate the path sensitizing method

When the test (01110) is applied, the output $Z_2 = 0$ in the absence of any fault at the outputs of $E_2, E_5, E_6$ and $E_8$ and $Z_2 = 1$ if any of these gates have their output s-a-1.

It can be observed from figure 2.2, that the paths $E_2E_4E_7$ and $E_2E_5E_6E_7$ are also sensitized, by the test (01110), to the output $Z_1$. However, the fault $E_2$ s-a-1 cannot be detected at $Z_1$ when the test (01110) is applied.

If $E_2$ is s-a-1, then, $E_4=0$ and $E_6=1$, giving $Z_1=1$.

If $E_2$ is not faulty, then, $E_4=1$ and $E_6=0$, giving $Z_1=1$. We see that there is no change in the output $Z_1$ due to the fault $E_2$ s-a-1. Hence, the test (01110) cannot detect the fault $E_2$ s-a-1 at $Z_1$. This occurs because the reconverging fan-out paths from $E_2$ to output $Z_1$ have unequal inversions. There is an inversion at gate $E_4$ in the path $E_2E_4E_7$, whereas, there is no inversion in the path $E_2E_5E_6E_7$.

In order to generate the complete set of tests, the path sensitizing method is used repeatedly, so as to cover all possible faults. For large networks such a test generation process is obviously lengthy. Also, the set of tests generate is not minimal.

(4) <u>d-Algorithm</u>:

This is a modification of the one-dimensional path sensitizing method, and was devised by Roth et.al.[17] in order to remove the discrepancies of the earlier method.

The idea of the d-algorithm is to simultaneously sensitize all possible paths from the site of the fault to the network outputs. The procedure is as follows:

(i) Pick a test for the fault $f_i$ in terms of the inputs and output of the failed gate.

(ii) Generate all possible paths simultaneously sensitized from the site of the fault to the network outputs.

(iii) Check for cancellation caused by reconvergent fan-out paths, and stop further consideration of the cancelled paths.

(iv) Try to assign values to the inputs, such that, all of the conditions imposed during the forward trace phase [step(ii)] are realized.

This procedure is repeated to cover all possible faults. As the size of the network increases, the number of possible faults also increases. Thus the number of iterations of the method increases. The set of tests that is generated by this procedure is not guaranteed as minimal. The path-generating method [10] presented in chapter 4 offers a more effective procedure to generate tests for all possible faults.

(5) Boolean-Difference Method:[19]

A DTS can be obtained directly from the Boolean expression of the functions realized by a network. The procedure

adopted to do so follows:

Consider a network of n inputs and let one of its outputs $Z_1$ be represented by,

$$Z_1 = \beta\ (x_1, x_2, \ldots, x_n)$$

where $\beta\ (x_1, x_2, \ldots, x_n)$ is a Boolean expression involving the inputs $x_1, x_2, \ldots, x_n$.

The Boolean-difference of $Z_1$ with respect to one of the inputs $x_i$ is denoted by $DZ_1(x_i)$ and is given by,

$$DZ_1(x_i) = \beta(x_1, x_2, \ldots, x_i, \ldots, x_n) \oplus \beta(x_1, x_2, \ldots, \bar{x}_i, \ldots, x_n)$$

where $\oplus$ is the exclusive-OR and $\beta(x_1, x_2, \ldots, \bar{x}_i, \ldots, x_n)$ is the expression $\beta(x_1, x_2, \ldots, x_i, \ldots, x_n)$ where all appearances of $x_i(\bar{x}_i)$ are replaced by the complement $\bar{x}_i(x_i)$. The new expression represents the output $Z_1$ when $x_i$ is at fault.

It can be easily verified that $DZ_1(x_i)$ is independent of $x_i$.

If $DZ_1(x_i) = 0$, it implies that $Z_1$ is independent of $x_i$.

If $DZ_1(x_i) = 1$, then any change in $x_i$ will affect the output $Z_1$ independent of the values of all $x_j$'s, $j \neq i$.

In general, $DZ_1(x_i)$ is a function of some or all $x_j$'s, $j \neq i$.

In order to detect a fault on input $x_i$, set $x_i$ opposite to the fault being tested for, that is, s-a-0 or s-a-1. Also, assign values to all other $x_j$'s, such that, $DZ_1(x_i) = 1$. The input combination resulting is a test for a fault on $x_i$, at the output $Z_1$.

$DZ_1(x_i).x_i$ is a test for $x_i$ s-a-0

and $DZ_1(x_i).\bar{x}_i$ is a test for $x_i$ s-a-1

The following example illustrates the Boolean-difference method.

Example:

The Boolean expression of the function realized by a network of 3 inputs at the output $Z_1$ is given by

$$Z_1 = x_1.x_2+\bar{x}_1.\bar{x}_2+\bar{x}_1.\bar{x}_3$$

Replacing $x_1$ by $\bar{x}_1$ and $\bar{x}_1$ by $x_1$, we get a new Boolean expression

$$Z_1(\bar{x}_1) = \bar{x}_1.x_2+x_1.\bar{x}_2+x_1.\bar{x}_3$$

Therefore, $DZ_1(x_1) = Z_1 \oplus Z_1(\bar{x}_1) = \bar{x}_2+x_3$

Similarly,

$$DZ_1(x_2) = x_1+x_3$$

and $$DZ_1(x_3) = \bar{x}_1.x_2$$

The test set is given in table 2.2.

| Faults | $x_1$ | $x_2$ | $x_3$ |
|---|---|---|---|
| $x_1$ s-a-1 | 0 | 0 | φ* |
| | 0 | φ | 1 |
| $x_1$ s-a-0 | 1 | 0 | φ |
| | 1 | φ | 1 |
| $x_2$ s-a-1 | 1 | 0 | φ |
| | φ | 0 | 1 |
| $x_2$ s-a-0 | 1 | 1 | φ |
| | φ | 1 | 1 |
| $x_3$ s-a-1 | 0 | 1 | 0 |
| $x_3$ s-a-0 | 0 | 1 | 1 |

Table 2.2 Test set for $Z_1=x_1x_2+\bar{x}_1\bar{x}_2+\bar{x}_1\bar{x}_3$

* φ denotes a don't care term..

By proper assignment of values to the don't care terms a smaller set of tests is obtained.

For the given example, the test set in table 2.3 is minimal.

| Faults | $x_1$ | $x_2$ | $x_3$ | $Z_1$ | Tests |
|---|---|---|---|---|---|
| $x_1$ s-a-1, $x_2$ s-a-0 and $x_3$ s-a-0 | 0 | 1 | 1 | 0 | $t_1$ |
| $x_1$ s-a-0 and $x_2$ s-a-1 | 1 | 0 | 0 | 0 | $t_2$ |
| $x_3$ s-a-1 | 0 | 1 | 0 | 1 | $t_3$ |

Table 2.3 Minimal test set

The DTS is represented by a diagnosing tree in figure 2.3.

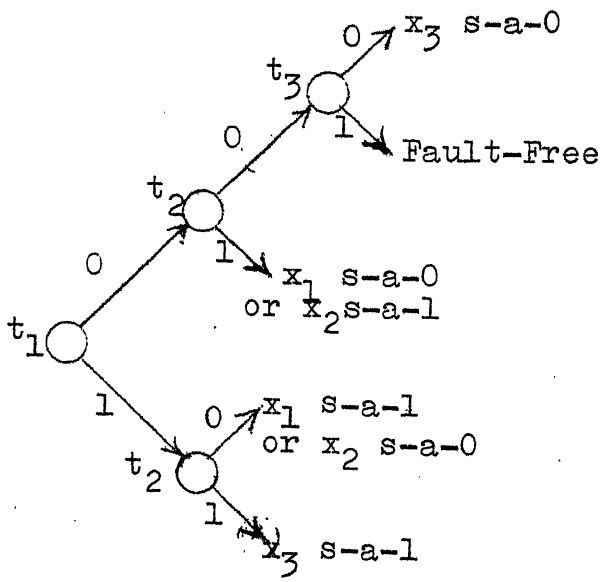



Figure 2.3 Diagnosing tree representation of the DTS

The Boolean-difference method is advantageous only in case of networks where the number of inputs is less than 100. As the number of inputs increases the manipulation of the Boolean expressions becomes unwieldy.

Summary:

In this chapter the various methods of generating tests, for detecting and locating faults in combinational networks, were presented. Of all these methods, the path sensitizing method has found the widest application, it is being employed in the diagnosis procedure of the IBM-360 series computers.

In the next chapter, graph-theory methods, for generation of DTS and for improvement of the diagnosis procedure, are studied.

# CHAPTER 3

## GRAPH-THEORY IN FAULT DIAGNOSIS

The structural and behavioural properties of a combinational network can be represented as digraphs. Here we are concerned with the use of the digraph models in the fault diagnosis of the network. The diagnostic procedure consists of the following:

(1) The analysis of the network graph, in which the nodes represent the smallest replaceable functional elements (single gates or modules) [13,18] and edges represent the interconnection between functional elements.

(2) The analysis of the behavioural aspects of the network.

(3) The derivation of test procedures based on (1) and (2).

For the fault diagnosis of large networks, the methods described in Chapter 2 have the following disadvantages.

(1) The time taken to generate a set of tests and to test the network is unduly large.

(2) The fault may not be located to the smallest replaceable element, that is, the diagnosibility is poor.

By taking into account the structural properties of the network the following can be achieved.

(1) Testing time is reduced, as the network can be segmented and the segments are tested in parallel.

(2) Means are provided to improve the diagnosibility of the network, for example, additional input and output points.

In Sec.3.1 we study the structural model of the network.

In Sec.3.2 the presently available methods for test generation using a graph-theoretic model are studied.

## 3.1 Structural Models

### Test-point approach [11,18]

By definition, a fault in the network can be detected if its presence produces a change in the value of atleast one of the network outputs. Therefore, as the number of network outputs increases, the probability of detecting all possible faults also increases. Also, any particular fault can be detected at more network outputs. Thus an increase in the number of network outputs improves the diagnosibility of the network and reduces the length of the DTS.

The number of network outputs can be artificially increased by breaking open the connections between the functional elements. Such break points are termed as test-points. Facilities are provided, to apply tests and monitor the outputs, at these test-points. At any time a test-point can be used as an input or an output point. When a set of input combinations is fed at one test-point (input point) and the output monitored at another test-point (output point), then

all the functional elements in the range of the two test-points are tested. The range of two test-points is the set of functional elements (nodes) lying on a directed path from the input to the output point.

Test-points are not required to be placed on all the interfconnections. A minimal set of test-points to cover all the functional elements, is obtained by the methods of [11]. We shall illustrate the procedure to obtain a minimal set of test-points, for fault-detection by an example.

Consider the digraph shown in figure 3.1, this represents a portion of a large network. The portion shown has 8 functional elements. Test-points are placed on the incoming edges to nodes 1 and 2 and on the outgoing edges from nodes 6,7 and 8.

The range-matrix R is now constructed from the adjacency matrix of the digraph [11]. For the given example, R is illustrated in table 3.1.

R (i,j) = 1 if the jth functional element is in the range of the ith test-point pair.

Otherwise, R (i,j) = 0

For the given example, two test-point pairs ($T_1$-$T_3$) and ($T_2$-$T_4$) are sufficient to cover all the nodes.

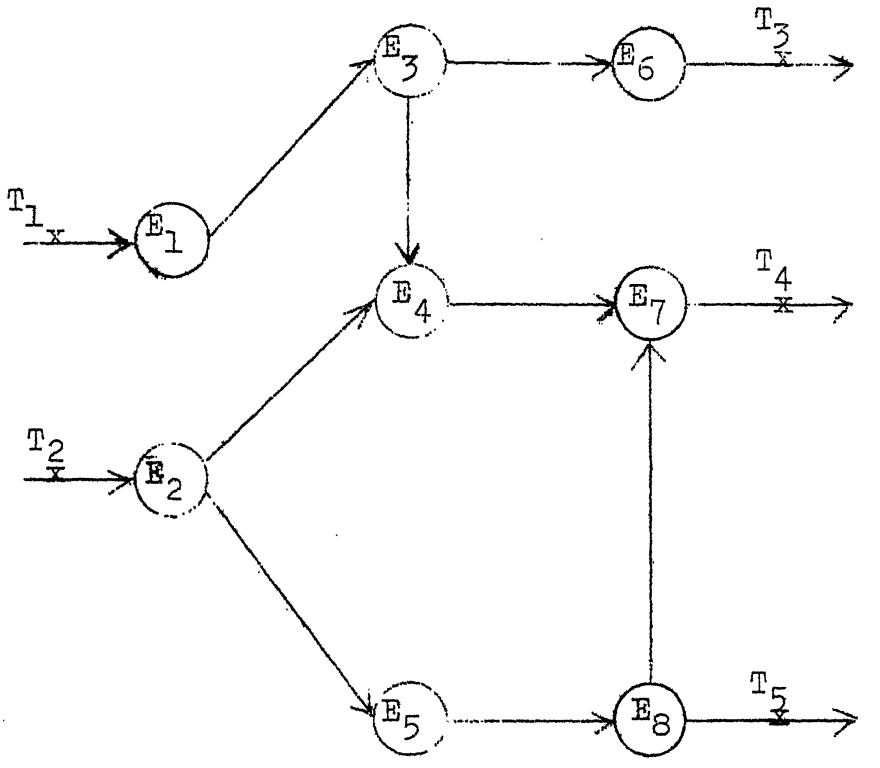



Figure 3.1 Network digraph

R =

| Test-Point Pairs | Functional elements | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $E_1$ | $E_2$ | $E_3$ | $E_4$ | $E_5$ | $E_6$ | $E_7$ | $E_8$ |
| $T_1$-$T_3$ | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| $T_1$-$T_4$ | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| $T_1$-$T_5$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $T_2$-$T_3$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $T_2$-$T_4$ | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| $T_2$-$T_5$ | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |

Table 3.1 Range-matrix R

The structural model can be employed to give the precedence-partitions, that is, the order in which the functional elements are to be tested. The precedence-partitions depend on the sequence in which information signal flows through the network. The following is the procedure to obtain the precedence-partitions from the adjacency matrix X of the digraph [11].

(1) The nodes corresponding to the columns of X having only zeros form the first precedence-partition $S_1$. Delete the rows and columns of X corresponding to the set of nodes in $S_1$.

(2) In the reduced adjacency matrix, repeat (1), to form $S_i$ ($i = 2, \ldots, u$), till all nodes are covered.

(3) $S_i$ ($i = 1, \ldots, u$) are the precedence-partitions of the network.

One testing scheme consists of testing all the precedence-partitions in parallel. In this scheme $u$ sets of tests are required, where $u$ is the number of precedence-partitions. Though the time taken in this scheme of testing is small, the disadvantage lies in the increased number of test-points. A serial testing scheme will require less number of test-points. In this scheme, tests are applied to the set of nodes in $S_1$ and output observed at the outputs of

a precedence-partition $S_i$ obtained by the following procedure.

(1) Let $P_1, P_2, \ldots, P_u$ be the known probability of failure of the functional elements in the precedence-partitions $S_1, S_2, \ldots, S_u$.

If the failure probabilities are not known, we can assume $P_1 = P_2 = \ldots = P_u$

(2) Find a precedence-partition $S_j$, such that,

$$(P_1+P_2 + \ldots + P_j) \geq (P_{j+1} + \ldots + P_u)$$

and $$(P_1+P_2 + \ldots + P_{j-1}) < (P_j+P_{j+1} + \ldots + P_u)$$

Test $S_j$, that is, test the network using the inputs at $S_1$ and the outputs of $S_j$.

If a fault is detected then it must be in a precedence-partition earlier in precedence to $S_{j+1}$, that is, in $S_1 \ldots S_j$.

Go to step 3.

If a fault is not detected, then we must test for $S_{j+1} \ldots S_u$.

Go to step 4.

(3) Find the precedence-partition $S_{j1}$, such that,

$$(P_1+P_2 + \ldots + P_{j1}) \geq (P_{j1+1} + \ldots + P_j)$$

and $$(P_1+P_2 + \ldots + P_{j1-1}) < (P_{j1} + \ldots + P_j)$$

Test $S_{j1}$. If a fault is detected, repeat step 3 for precedence-partitions $S_1 \ldots S_{j1}$.

If a fault is not detected, then we must test for $S_{j1+1} \ldots S_j$.

Go to step 4.

(4) Find $S_{j2}$ such that,

$$(P_{j+1} + \ldots + P_{j2}) \geqslant (P_{j2+1} + \ldots + P_u)$$

$$\text{and} (P_{j+1} + \ldots + P_{j2-1}) < (P_{j2} + \ldots + P_u)$$

Test $S_{j2}$. If a fault is detected,

test for $S_{j+1} \ldots S_{j2}$

Go to step 3.

If a fault is not detected, we must test for $S_{j2+1} \ldots S_u$. Repeat step 4.

The procedure is repeated until all the precedence-partitions are uniquely identified.

We shall illustrate the procedures described here by an example.

Example: Consider the network shown in figure 3.2. The structural model for this network is given in figure 3.3 and the adjacency matrix in table 3.2. From the adjacency matrix we find the precedence-partitions. For the given example these are:

$$S_1 = (E_1, E_3)$$

$$S_2 = (E_2, E_4)$$

$$S_3 = (E_5, E_6, E_7)$$

Consider the failure probabilities of $S_1$, $S_2$ and $S_3$ as same. Hence, using the procedure described earlier,

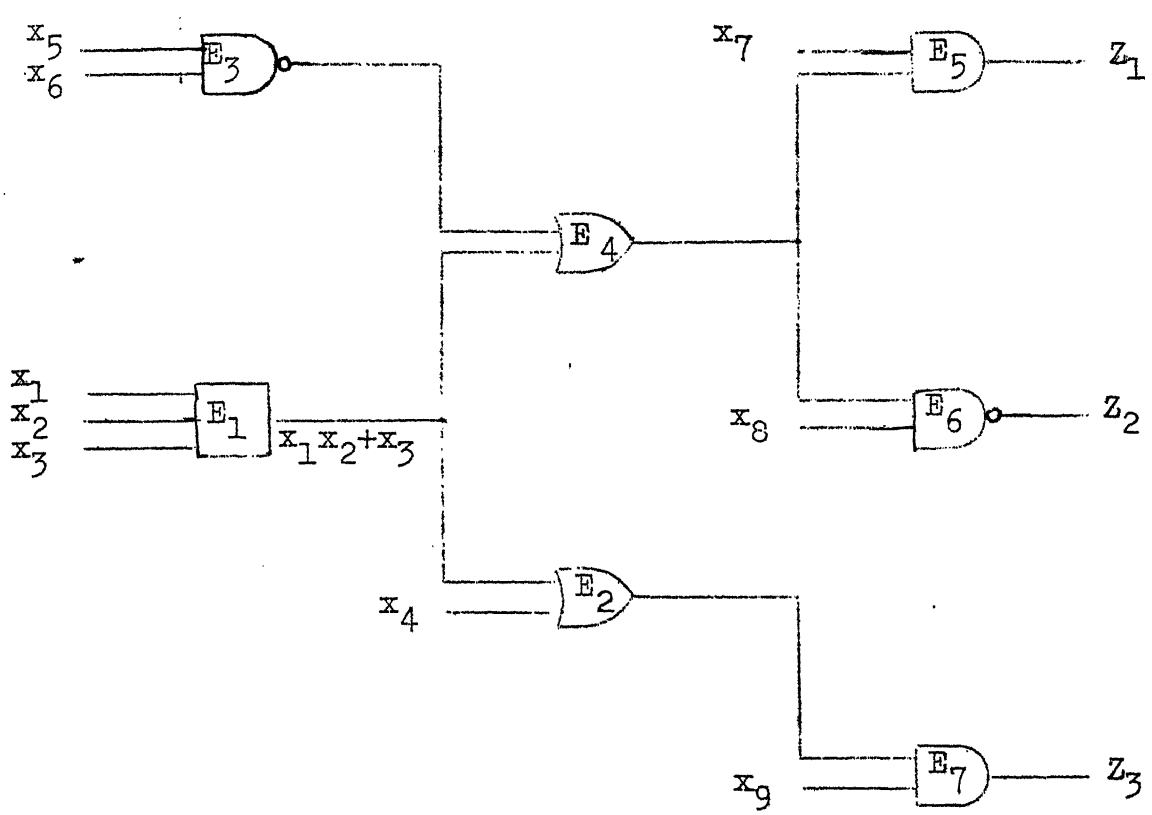



Figure 3.2 Example network

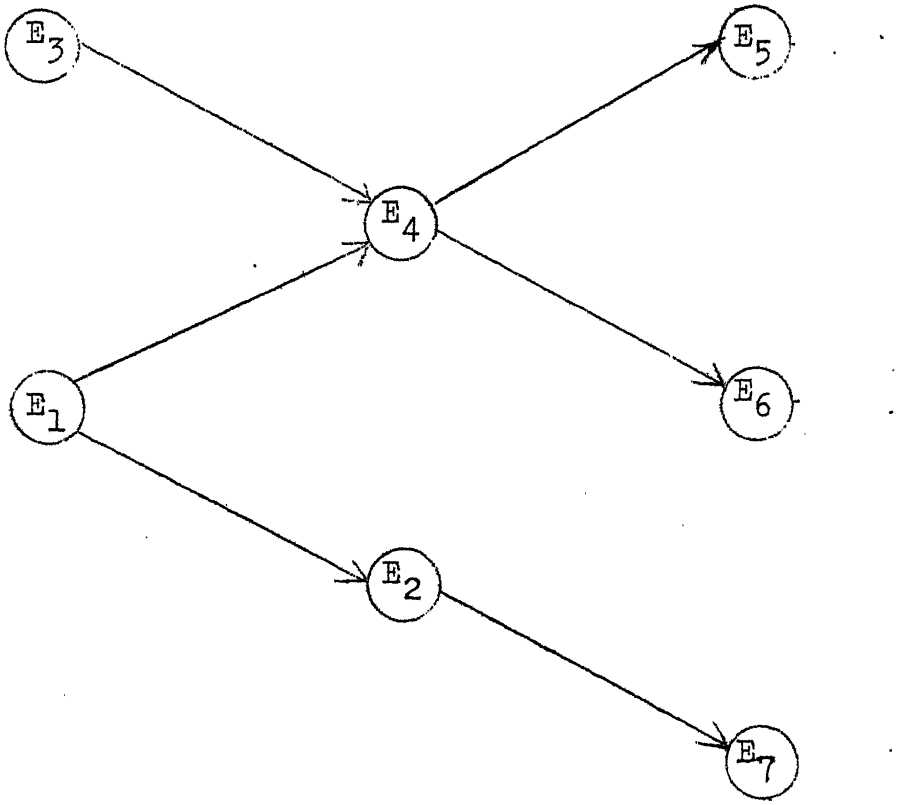



Figure 3.3 Structural model of the network in figure 3.2

X =

| | $E_1$ | $E_2$ | $E_3$ | $E_4$ | $E_5$ | $E_6$ | $E_7$ |
|---|---|---|---|---|---|---|---|
| $E_1$ | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| $E_2$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| $E_3$ | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| $E_4$ | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| $E_5$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $E_6$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $E_7$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Table 3.2 Adjacency matrix of digraph in figure 3.3

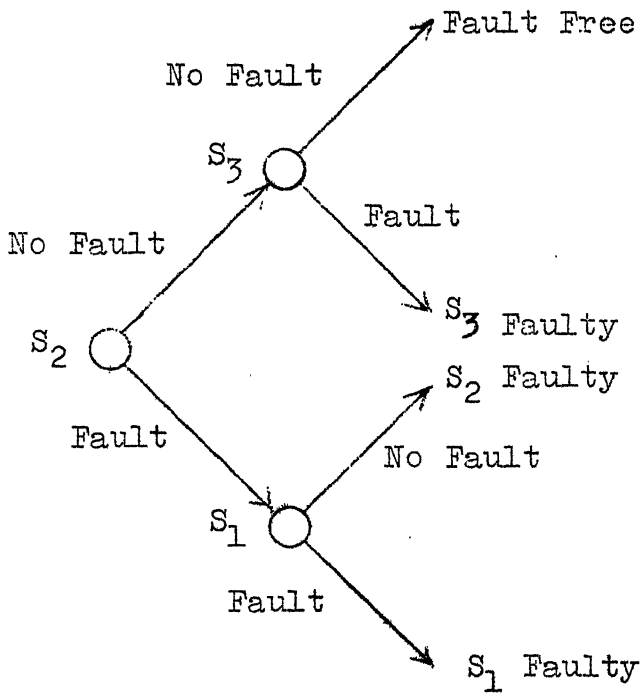



Fig.3.4 Serial testing sequence of the precedence-partitions

we obtain,

$$(P_1 + P_2) \geq (P_3)$$

and $$(P_1) < (P_2 + P_3)$$

Therefore, first test $S_2$. If a fault is detected, test $S_1$.

If a fault is not detected, test $S_3$.

The sequence of testing is given as a diagnosing tree in figure 3.4.

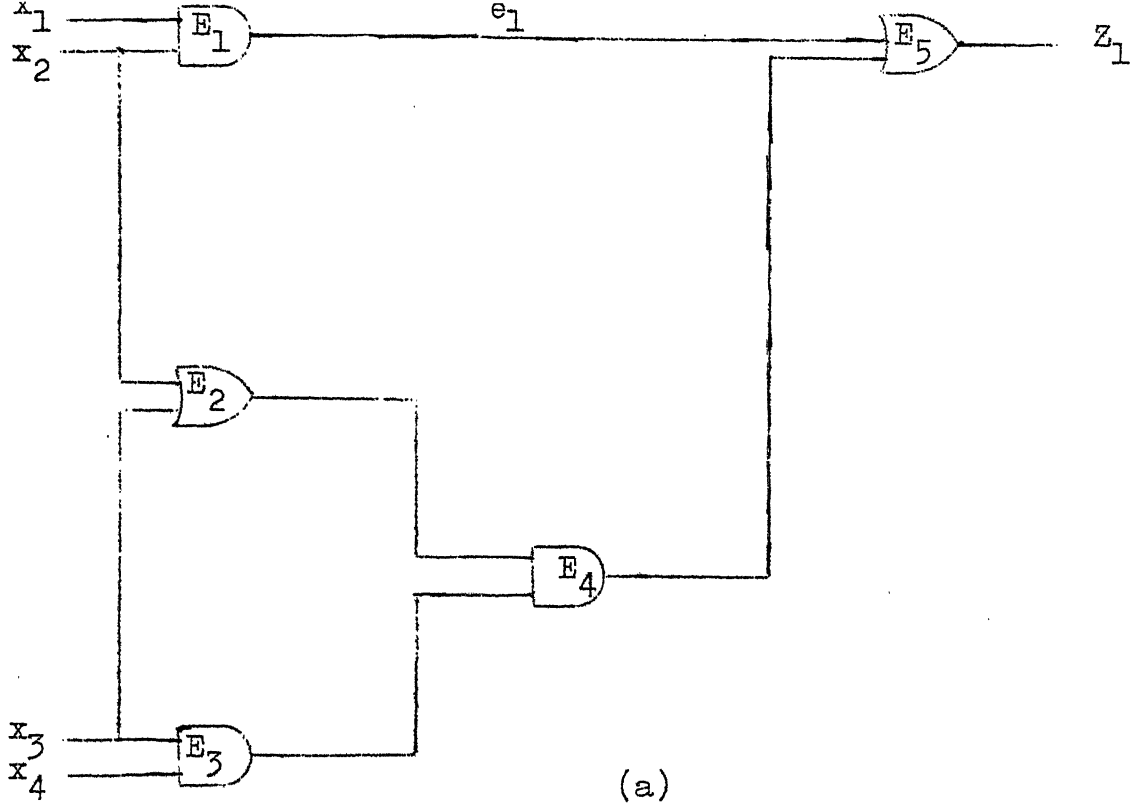
Since test-points segment the network, the effect of a fault in one portion of the network is not transferred to an adjacent portion. Thus, the single-faults are not allowed to be converted into multiple-faults, and hence they can be readily detected.

A testing scheme using the test-point approach requires additional hardware and facilities to break open the interconnections between elements. A separate hardware tester has to be provided for applying the tests. Thus, the cost of using this approach is obviously more than the other methods.

Blocking-gates and diagnosis [15]

In chapter 2 we discussed the path-sensitizing method where the fault is detected by sensitizing a path from the site of the fault to one of the network outputs. A similar approach using the digraph model of the network is described in this section.

A blocking-gate is a single logic gate or a combination of gates. When placed on any edge it can be controlled to block or deblock that edge. In the simplest case the blocking of an edge would be equivalent to imposing a signal state of a logical 0 or a logical 1 on the output end of that edge. Thus, an AND or an OR gate would be sufficient. In practical cases a blocking-gate on an edge will consist of a number of gates. The cost of implementing a blocking-gate varies and depends entirely on the edge on which it is located and the function (behavioural characteristics) of the nodes associated with the edge. Two examples of implementation are indicated in figure 3.5. In the first example, figure 3.5(a), the effect of blocking signals through an edge can be controlled by proper choice of inputs. The input signal $\bar{x}_1$ has the same effect as the blocking of edge $e_1$. In the second example, figure 3.5(b), more than one gate is required to block the edge $e_2$. An OR gate $E_3$ with the control input $c_1$ set at 1 and an AND gate $E_4$ with the control input $c_2$ set at 0, blocks the edge $e_2$.



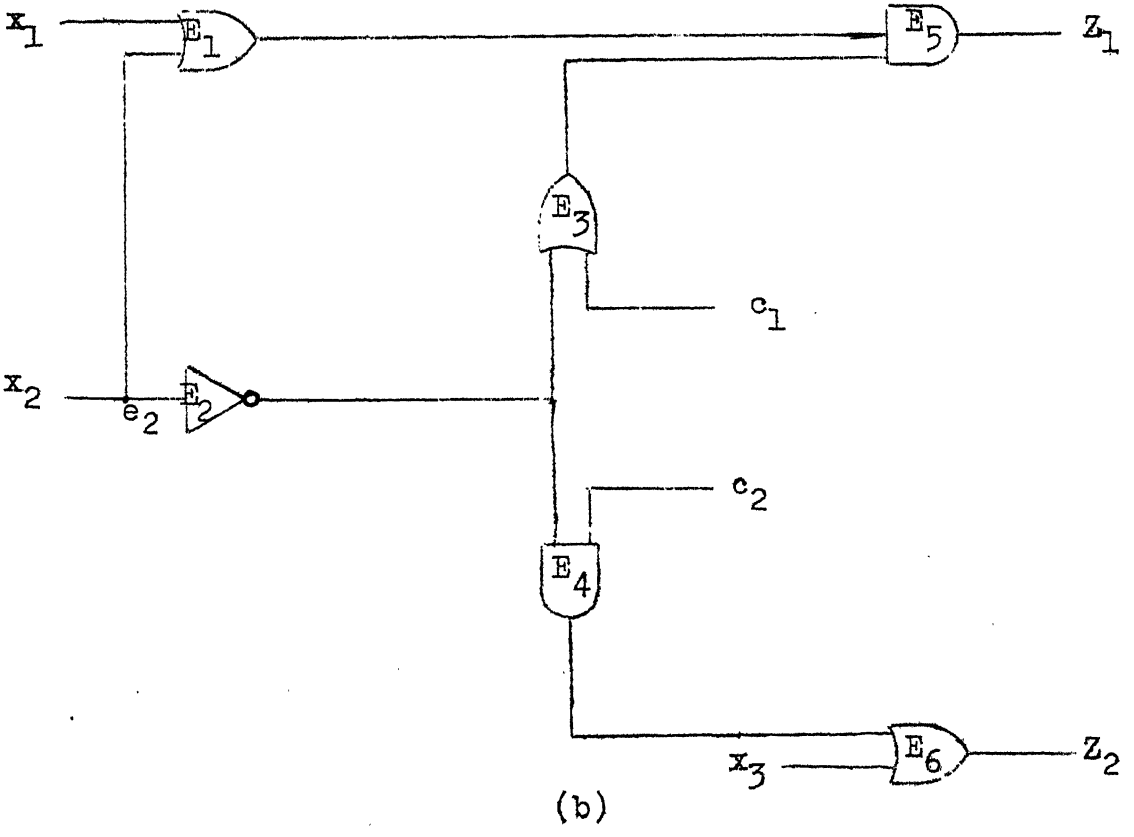




Figure 3.5 Example to illustrate placement of blocking-gates

In the blocking-gate approach, tests are applied to the network inputs and the outputs monitored at the network outputs. The effect of sensitizing a path from a functional element to one of the network outputs is achieved by simply blocking the edges of the other paths. In contrast to the test-point approach, no additional input or output points are required in the blocking-gate approach. The placement and control of blocking-gates has been observed to be less costly than the use of test-points [15].

The improvement in the diagnosibility of the network by using blocking-gates is illustrated by an example.

Example Consider the two digraphs shown in figure 3.6. G is a single-entry single-exit connected (SEC)* graph from $E_i$ to $E_o$. $G_1$ is a subgraph of G and it also is a SEC graph from $E_i$ to $E_o$.

If there is a faulty node in G, it can be located by applying tests at $E_i$ and observing the output at $E_o$. Now, subgraph $G_1$ is tested by blocking the edge $e_6$. If no fault is detected by testing $G_1$, the faulty node is obviously $E_1$ and is given by,

$$V(G) \cap V(G_1)' = (E_1)$$

where,

$V(G)$ = The set of all nodes in G

$V(G_1)$ = The set of all nodes in $G_1$

$V(G_1)'$ = The set of all nodes in G but not in $G_1$.

If the actual network has multiple inputs or outputs, we add a dummy input or output node and connect them to the actual inputs or outputs, so that the graph has only one

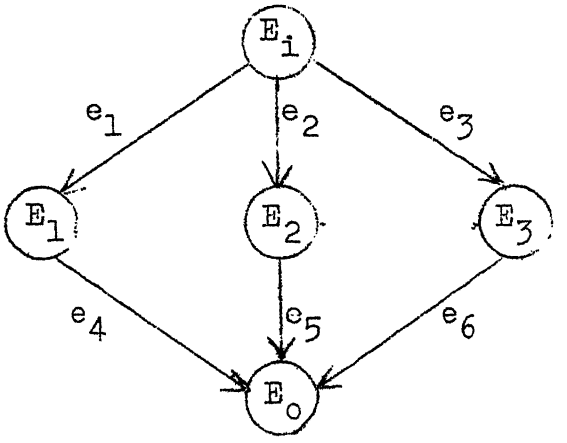



Figure 3.6(a) SEC graph G

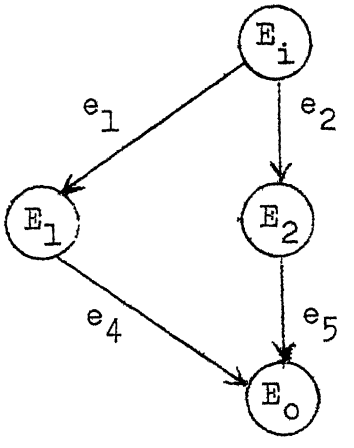



Figure 3.6(b) Subgraph $G_1$

From the given example, we see that by blocking some edges, subgraphs of G can be tested and the faulty node easily located.

If, for all possible permutations of blocked edges of the SEC graph, two nodes $E_a$ and $E_b$ are both either in the subgraphs formed or not in them, then the faults in these nodes are indistinguishable. Because, in no case is there a path from $E_i$ to $E_o$ which contains only one of $E_a$ or $E_b$.

Blocking-gates are not required to be placed on all the edges. An algorithmic procedure given in [15] determines the placement of blocking-gates for maximum distinguishability. We shall illustrate the procedure for the network given in figure 3.2. The SEC graph of the network is illustrated in figure 3.7.

The procedure follows:

(1) In the SEC graph from $E_i$ to $E_o$ find the range of each node. The range of a node $E_a$ is the union of nodes of all paths from $E_i$ to $E_o$ after deleting $E_a$.

(2) Form the node-range matrix M.

$M(a,b) = 1$ if the node $E_b$ is in the range of node $E_a$.

Otherwise, $M(a,b) = 0$ .

For the given example M is illustrated in table 3.3.

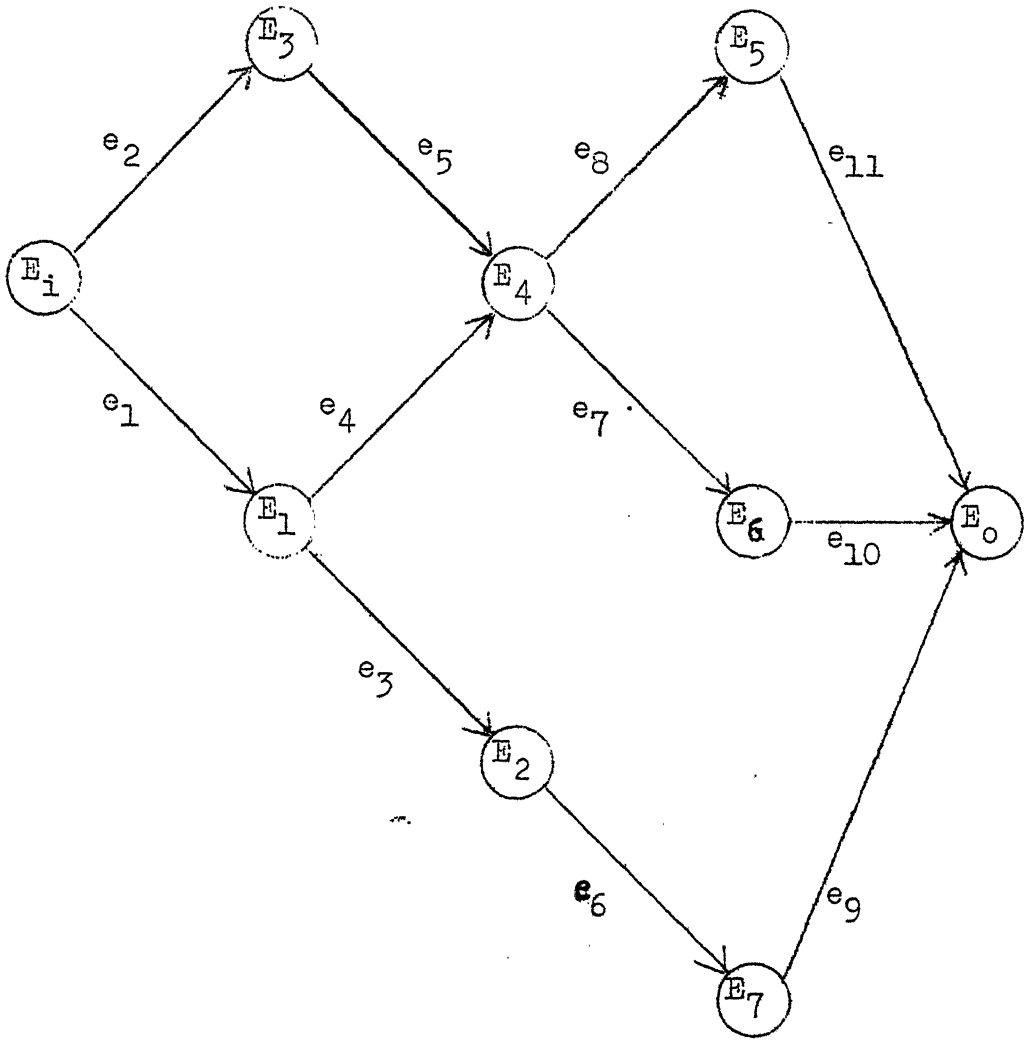



Figure 3.7 SEC graph of the network in figure 3.2

M =

| | $E_1$ | $E_2$ | $E_3$ | $E_4$ | $E_5$ | $E_6$ | $E_7$ |
|---|---|---|---|---|---|---|---|
| $E_1$ | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| $E_2$ | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| $E_3$ | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| $E_4$ | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| $E_5$ | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| $E_6$ | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| $E_7$ | 1 | 0 | 1 | 1 | 1 | 1 | 0 |

Table 3.3 Node-range matrix of the SEC graph

(3) The nodes corresponding to identical columns of M are indistinguishable. Each set of indistinguishable nodes is termed as a D-partition.

For the given example,

$D_1 = (E_1)$

$D_2 = (E_2, E_7)$

$D_3 = (E_3)$

$D_4 = (E_4)$

$D_5 = (E_5)$

$D_6 = (E_6)$

Here, the faults in $E_2$ and $E_7$ are indistinguishable. The elements in each D-partition may be packed together in a module, for ease in replacement.

(4) Form the edge-range matrix Y. Each row of Y corresponds to the range of an edge. The range of an edge $e_a$ is the union of nodes of all paths from $E_i$ to $E_o$ after deleting edge $e_a$. For the given example Y is illustrated in table 3.4.

(5) The reduced edge-range matrix is obtained by replacing each set of identical rows of Y by a single row. The reduced edge-range matrix for the given example is illustrated in table 3.5.

(6) From the reduced Y matrix, the optimal set of edges on which blocking-gates are to be placed is obtained. This can be achieved by using the method given in chapter 2 for

| | | D-partitions | | | | | |
|---|---|---|---|---|---|---|---|
| | Edges | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ |
| | $e_1$ | 0 | 0 | 1 | 1 | 1 | 1 |
| | $e_2$ | 1 | 1 | 0 | 1 | 1 | 1 |
| | $e_3$ | 1 | 0 | 1 | 1 | 1 | 1 |
| | $e_4$ | 1 | 1 | 1 | 1 | 1 | 1 |
| | $e_5$ | 1 | 1 | 0 | 1 | 1 | 1 |
| Y = | $e_6$ | 1 | 0 | 1 | 1 | 1 | 1 |
| | $e_7$ | 1 | 1 | 1 | 1 | 1 | 0 |
| | $e_8$ | 1 | 1 | 1 | 1 | 0 | 1 |
| | $e_9$ | 1 | 0 | 1 | 1 | 1 | 1 |
| | $e_{10}$ | 1 | 1 | 1 | 1 | 1 | 0 |
| | $e_{11}$ | 1 | 1 | 1 | 1 | 0 | 1 |

Table 3.4 Edge-range matrix Y of the SEC graph in figure 3.7

| | | D-partitions | | | | | |
|---|---|---|---|---|---|---|---|
| | Edges | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ |
| | $e_1$ | 0 | 0 | 1 | 1 | 1 | 1 |
| | $e_2$ | 1 | 1 | 0 | 1 | 1 | 1 |
| | $e_3$ | 1 | 0 | 1 | 1 | 1 | 1 |
| Y = | $e_4$ | 1 | 1 | 1 | 1 | 1 | 1 |
| | $e_7$ | 1 | 1 | 1 | 1 | 1 | 0 |
| | $e_8$ | 1 | 1 | 1 | 1 | 0 | 1 |

Table 3.5 Reduced edge-range matrix

finding the optimal set of tests from the fault-matrix. For the given example, blocking-gates are to be placed on the set of edges--$(e_1, e_2, e_3, e_7, e_8)$.

(7) The sequence in which the edges are blocked and tests applied is shown as a diagnosing tree in figure 3.8.

The blocking of the edges in $(e_1, e_2, e_3, e_7, e_8)$ is achieved in the following manner.

Edge $e_1$ is blocked by setting $x_1 = x_3 = 0$

Edge $e_2$ is blocked by setting $x_5 = x_6 = 1$

Edge $e_3$ is blocked by placing an AND gate $E_8$ on $e_3$ and setting control $c_3 = 0$

Edge $e_7$ is blocked by placing an OR gate $E_9$ on $e_7$ and setting control $c_7 = 1$

Edge $e_8$ is blocked by placing an OR gate $E_{10}$ on $e_8$ and setting control $c_8 = 1$.

The network, of the given example, with blocking-gates, is illustrated in figure 3.9.

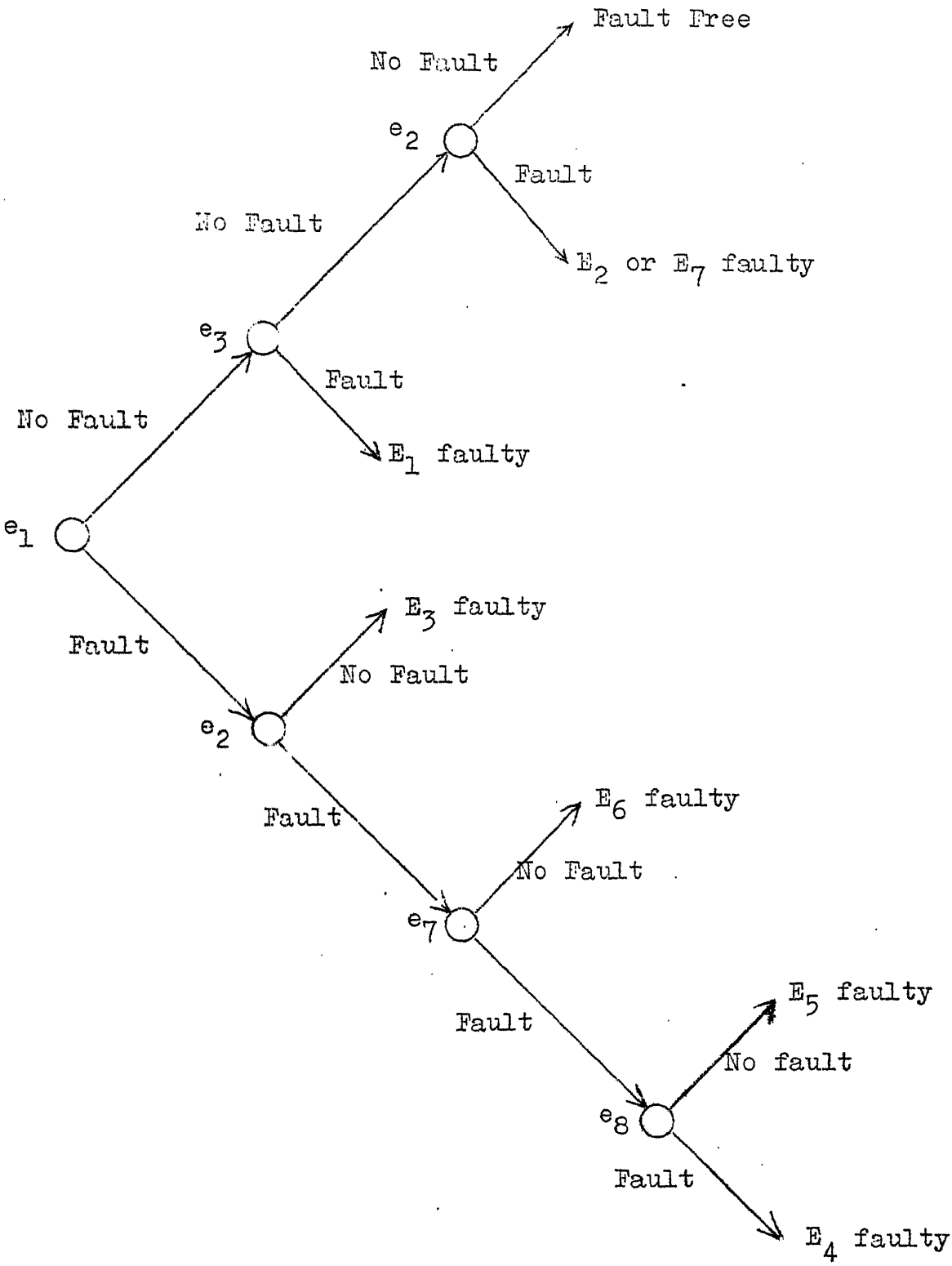



Figure 3.8 Diagnosing tree representing the order in which edges are blocked and tests applied.

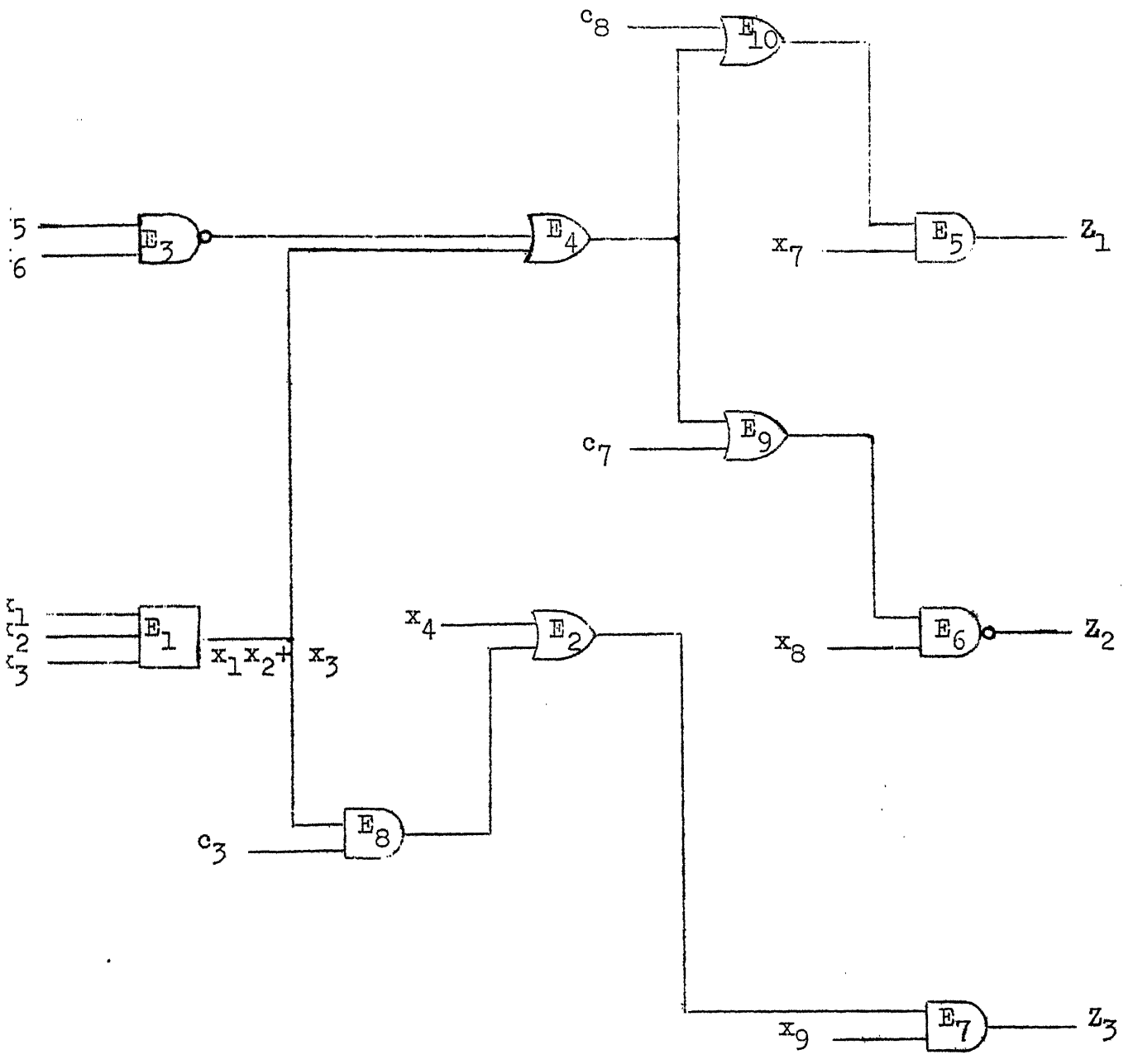



Figure 3.9 The network of figure 3.2 with blocking-gates $E_8, E_9, E_{10}$

## 3.2 Behavioural Models

### Fault diagnosis of single-gate faults

Hornbuckle and Spann [8] have devised two methods for fault diagnosis of single-gate faults. Memory and time bounds of the test sets are also given.

Assumption of only a single-gate being faulty eliminates the need for enumerating all possible faults and the construction of a fault-matrix. Both the methods given in [8] require a set of tests to sensitize each gate to one of the network outputs. Each gate is sensitized for all the $2^p$ input combinations at its terminals, where p = number of input lines to the gate. Such a set of tests will detect all possible faults of each gate.

A diagnostic test set to distinguish between each pair of possibly faulty gates, is also determined.

The upper bound on the number of tests for fault detection and location is given in expression 3.1.

$$K.2^p + (K-1) \qquad (3.1)$$

Where, K = number of gates in the network.

The assumption made in [8], whilst deriving the upper bound given in expression 3.1, is that all gates have the same number of input lines. In most practical cases this is not valid. However, if we assume that p represents the maximum number of input lines to any gate, then expression 3.1

is valid as the worst case upper bound. In most cases the number of tests for fault detection and location will be much lower than the upper bound.

In the second method, a bipartite graph is constructed for each gate-pair. The nodes in this graph represent the input combinations at a gate and also indicate whether the gate is sensitized to a network output by a test. The edges represent the tests, There is an edge between two nodes if the corresponding test causes the indicated conditions at the two gates. An example of such a graph for two gates $E_a$ and $E_b$ is illustrated in figure 3.10.

A reduced graph is formed, illustrated in figure 3.11, where all unsensitized nodes are merged into a single sink node. The tests corresponding to the edges of a tree of each of the reduced graph components constitutes a test-set for the gate-pair. A test set is determined for each gate-pair. In the worst case, the number of tests in the DTS for the network is given in expression 3.2

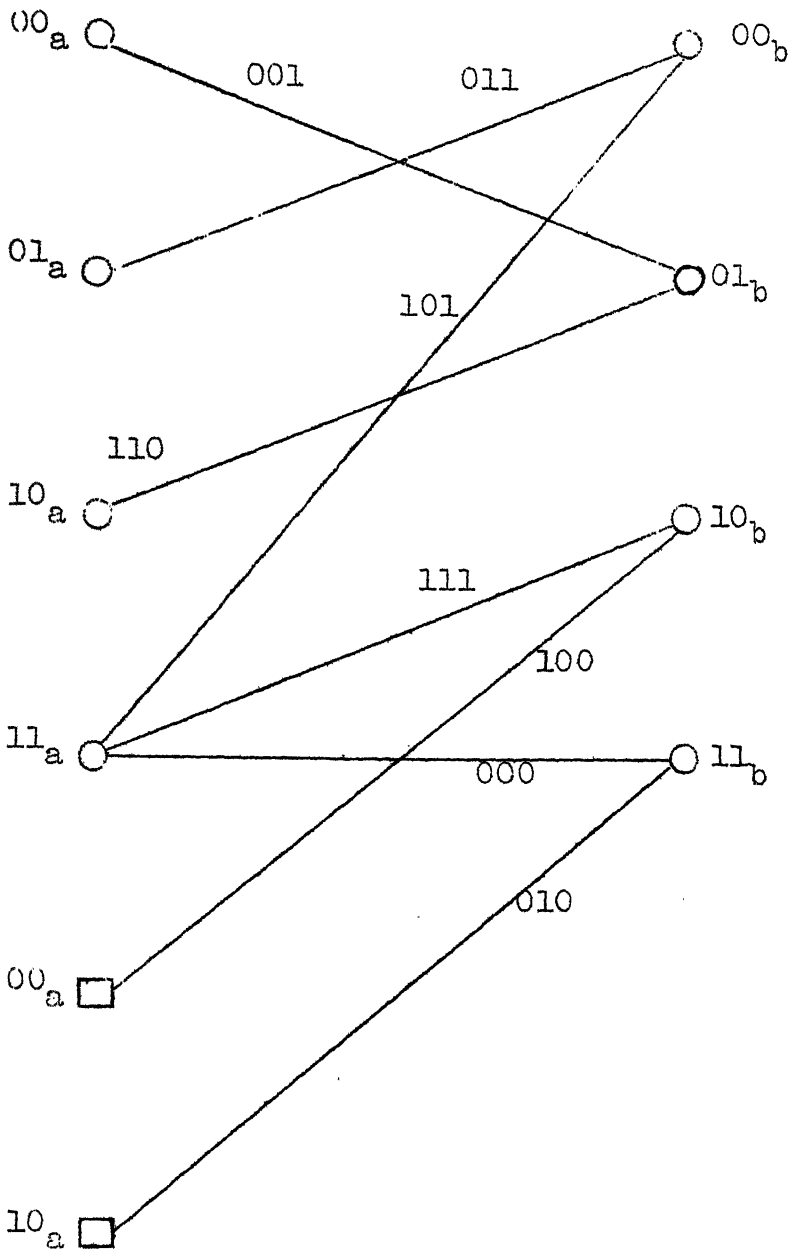
$$K.(K-1).\ 2^{p} \qquad (3.2)$$

Expression 3.2 is the upper bound on the memory required to store the DTS.

The testing procedure is as follows:

(1) Assign a value TRUE to the sink node.

(2) Consider a branch of the tree generated, connecting two




○ sensitized node

□ unsensitized node

p = 2, n=3

$ij_a$ = input combination at the terminals of gate $E_a$.

Fig.3.10 Complete graph for gate-pair $E_a$-$E_b$

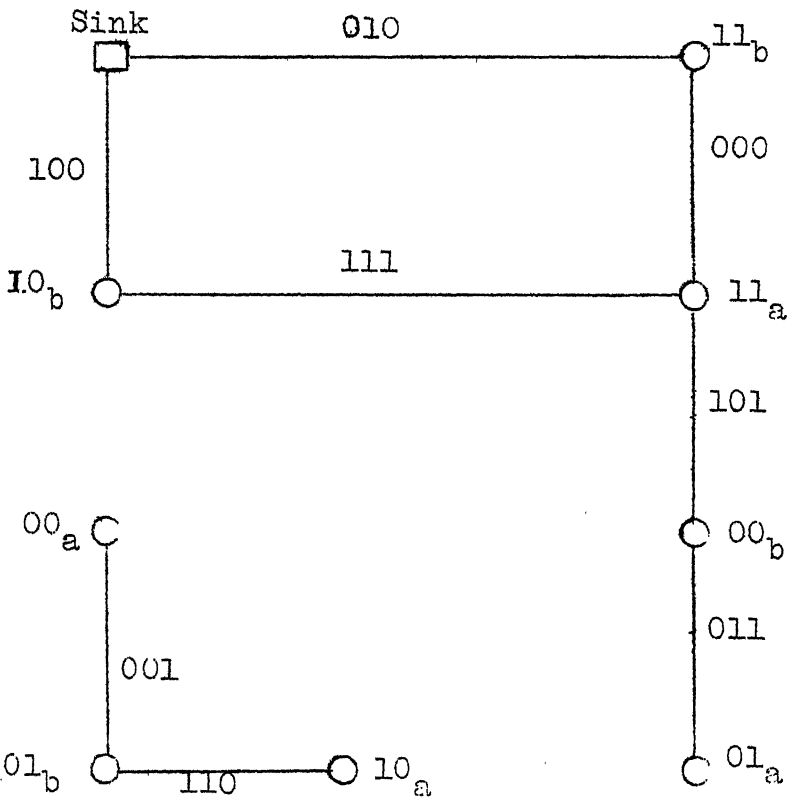



Fig.3.11 Reduced graph for gate-pair $E_a$-$E_b$

sensitized nodes. Apply the associated test. If the network outputs corresponding to this test are correct, assign the value TRUE to both nodes. Otherwise assign the value FALSE to both nodes.

(3) Apply the tests corresponding to each of the tree branches, one at a time. Observe the outputs and assign values to the corresponding nodes. If a node already has a value assigned to it, and the application of test assigns it an opposite value a conflict occurs. This causes the gate corresponding to that node to be eliminated. This is because, a gate cannot generate two different outputs for the same input combination at its input terminals. A violation terminates the testing for the gate-pair, as one of the two gates is eliminated.

(4) An error for a branch connecting the sink to a sensitized node, causes violation for the sink node, and the gate that does not correspond to the sensitized node is eliminated.

From the given testing procedure, we see that one gate is eliminated after the testing between each gate-pair. Hence, the upper bound on the number of tests to be applied is as in expression 3.3.

$$(K-1).\ 2^{p+1} \qquad (3.3)$$

Expression 3.3 is the upper bound on the testing time.

In [8] it is shown that " for complete diagnosis, the number of tests which must be applied to the network is proportional to the number of gates and is independent of the number of network input lines" . However,it must be noted, that the process of generating the DTS is very much dependent on the number of network input lines. This is because, all the $2^n$ input combinations have to be applied while checking for sensitized. paths from each gate to a network output. Also, checking for sensitized paths becomes lengthy and involved as the number of levels and gates in the network increases. Therefore, though the methods given in [8] provide reliable fault location, for large networks the generation of the DTS becomes cumbersome. The method proposed by Koga et.al. [10] is simpler and more convenient for generating the DTS.for large networks.

Path-generating method (PGM)

Carroll et.al. [3] have dealt with fault diagnosis of computers based on a modular construction and using LSI modules.

Koga et.al. [10] have extended the above [3] for test generation for combinational networks. The approach is similar to the path-sensitizing method. The network to be diagnosed is represented as a set of Boolean expressions, where complementation occurs only with the individual variables. A

digraph is then obtained from the expressions by translating logical sum (OR) and logical product (AND) into parallel and series connections, respectively. A set of paths is then generated (using the given PGM) from the digraph and its complement graph. It is proved that this set is irredundant and sufficient for detection and location of all distinguishable single s-a-0 and s-a-1 faults.

Actual test generation using the method presented in [10], has been done for the Processing Element control logic of the ILLIAC IV computer.

In the next chapter the PGM is presented in detail. A modification of this method is also suggested.

# CHAPTER 4

# TEST GENERATION FROM THE DIGRAPH MODEL OF A NETWORK

## 4.1 Conversion of logic description into a digraph

In the PGM [10] the network is described by the Boolean expressions of the functions realized. The Boolean expressions are processed such that the NOT (complement) operation is only on the individual variables. After this, the logical sum (OR) operations are translated into parallel connections and the logical product (AND) operations into series connections.

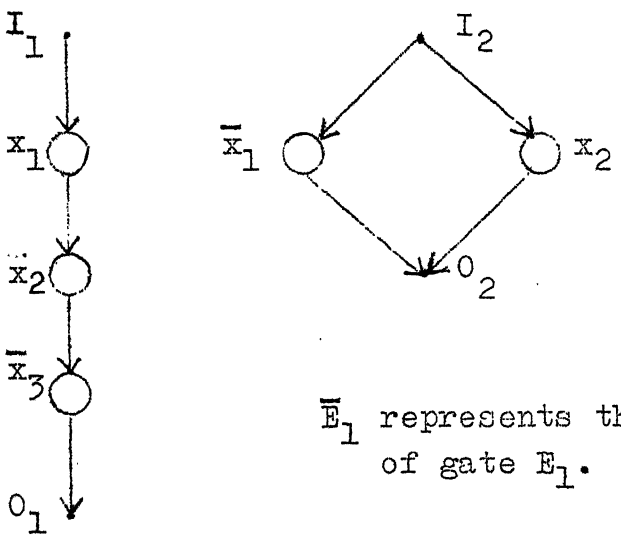
Here, we describe a method to obtain the digraph representation of the network directly from the wiring diagram of the network. The logic description of the network is described in the format of table 4.1.

After reading the description of each gate, its subgraph is formed. Consider the logic description of table 4.1. The type of gates are AND, OR and NOT. The subgraphs of the three gates are shown in figure 4.1. $I_i$ and $O_i$ represent the input and output nodes, respectively, of the subgraph of gate $E_i$. These are dummy nodes and are merged whenever two subgraphs are connected.

If one of the inputs to a gate $E_i$ is the output of another gate $E_j$, then, in the subgraph of $E_i$ we substitute

| GATE | TYPE | INPUTS | FAN-OUT | OUTPUTS |
|---|---|---|---|---|
| $E_1$ | AND | $x_1, x_2, \bar{x}_3$ | 2 | $E_3, Z_1$ |
| $E_2$ | OR | $\bar{x}_1, x_2$ | 1 | $Z_2$ |
| $E_3$ | NOT | $E_1$ | 1 | $Z_3$ |

Table 4.1 Example logic description




$\bar{E}_1$ represents the complement of output of gate $E_1$.

Figure 4.1 Subgraphs of gates $E_1$, $E_2$ and $E_3$.

the node representing output of $E_j$ by the subgraph of $E_j$.

For any network, the procedure described is repeated until the entire graph has nodes which represent the individual input variables.

In the logic description format shown in table 4.1, the fan-out entry for any gate $E_i$ is the number of gates to which the output of $E_i$ is fed. This gives a count of the number of times the subgraph of gate $E_i$ is to be substituted while forming the digraph of the network. Each time the subgraph of gate $E_i$ is substituted the fan-out is reduced by 1. When the fan-out becomes 0, the subgraph of gate $E_i$ is deleted. This affects a saving in memory requirement when the test generation is being done by a computer.

The complement graph of the network is now obtained from the original graph, by changing all parallel connections into series connections and all series connections into parallel connections. At the same time complementing each variable represented by a node in the original graph.

The subgraph of gate $E_3$, illustrated in figure 4.1, is modified by replacing the node $\bar{E}_1$ by the complement subgraph of gate $E_1$. This is illustrated in figure 4.2.

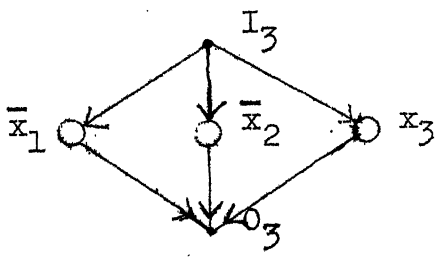



Figure 4.2 Subgraph of gate $E_3$

The original and complement graphs are connected to form a SEC graph. In the next section, the PGM of [10] is described.

## 4.2 Path-generating method

The PGM provides a set of paths from the input to the output node of the SEC graph representing the network. This set of paths is sufficient to cover all nodes in the SEC graph. And, it will provide tests to locate all distinguishable single s-a-0 and s-a-1 faults of the network.

The advantage of the PGM over those presented in chapter 2 is that here only one set of tests (paths) is generated and this is minimal and sufficient to locate all distinguishable faults, provided the network is irredundant and has a pure tree structure(i.e. no reconvergent or feedback path In any other method, for example, the path-sensitizing method, a test is generated for each fault, after which the minimal number of tests is determined.

The procedure to generate the set of paths as given in [10] follows:

(1) In the SEC graph, the input node is assigned a level 0.

(2) All nodes adjacent to the input node are assigned a level 1.

(3) All nodes adjacent to the level 1 nodes are then listed and assigned level 2.

I.I.T. KANPUR
CENTRAL LIBRARY
28384

(4) Step 3 is repeated until all nodes are covered. If a node has already occurred on a higher level or previously on the same level, it is termed a pseudo-terminal node, and path tracing from it down to the output node is stopped.

(5) Whenever a path from the input node reaches a pseudo-terminal node, the path is completed by arbitrarily choosing any route (usually the shortest) that goes from it to the output node.

Paths in which a conflict occurs, that is, both the node $x_i$ and its complement node $\bar{x}_i$ are present in the same path, are not considered.

To generate a test, the variables in one of the paths generated by the PGM are assigned values such that the path connects the input node to the output node. The remaining variables are assigned values such that no other path connects the input and output nodes.

A test-path in the original graph tests for all s-a-0 faults of the uncomplemented variables and all s-a-1 faults of the complemented variables, which are represented by nodes in the path. Each test-path in the original graph also tests for the output s-a-0.

A test-path in the complement graph tests for all s-a-1 faults of the complemented variables and all s-a-0 faults of the uncomplemented variables, which are represented by nodes

in the path. Each test-path in the complement graph also tests for the output s-a-1.

While assigning values to a particular test-path $t_i$, it may happen that another test-path $t_j$ is also activated. Such a test-path is a sneak-path and may be removed by changing the assignment of values of the two-sided variables outside the test-path $t_i$. A two-sided variable is one which occurs in both complemented as well as uncomplemented form in the original graph. If the sneak-path $t_j$ cannot be eliminated by changing the two-sided variables, then $t_i$ can test only the variables belonging to $t_i \cap t_j$.

Conflicts and sneak-paths occur in accordance with the structure of the network. In case of networks having a pure tree structure, conflicts are few. Sneak-paths occur because of redundancy in the network.

The PGM will therefore be most effective in case of irredundant networks having a pure tree structure. In cases where the network does not have a pure tree structure conflicts will occur reducing the effectivity of the PGM. This discrepancy is similar to that of the path-sensitizing method described in chapter 2. The reasons are same, that is, reconvergent paths with unequal inversions are present in the network.

An example illustrating the PGM and the modification proposed is given in the next section.

## 4.3 Example

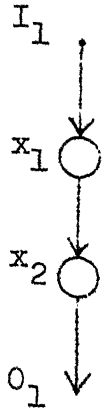
Consider the logic description given in table 4.2.

| Gate | Type | Inputs | Fan-out | Outputs |
|---|---|---|---|---|
| $E_1$ | AND | $x_1, x_2$ | 1 | $E_3$ |
| $E_2$ | NOT | $x_3$ | 2 | $E_3, E_5$ |
| $E_3$ | OR | $E_1, E_2$ | 1 | $E_4$ |
| $E_4$ | NOT | $E_3$ | 1 | $E_7$ |
| $E_5$ | AND | $x_4, E_2$ | 1 | $E_6$ |
| $E_6$ | OR | $x_5, E_5$ | 1 | $E_7$ |
| $E_7$ | OR | $E_4, E_6$ | 1 | Z |

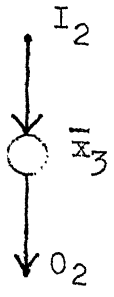
Table 4.2 Logic description of example network

The subgraph of each gate is now formed and then the complete SEC graph representation of the network is obtained. The method described in sec.4.1 is employed, the complete process is described in figures 4.3-4.5.

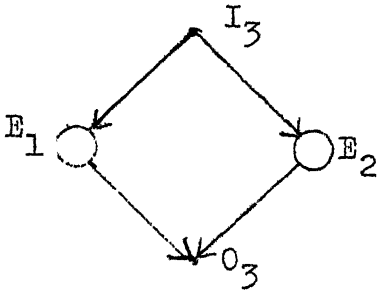
The method of [10] given in sec.4.2 is now used to generate test-paths from the input to the output node of the SEC graph. The generated paths are shown in figure 4.6.



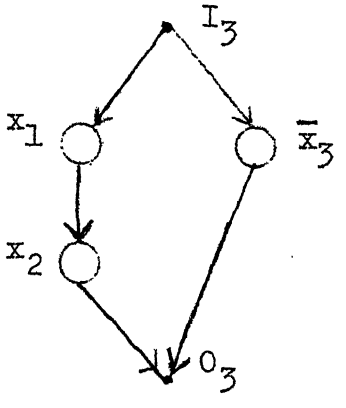

Step 1: Subgraph of gate $E_1$



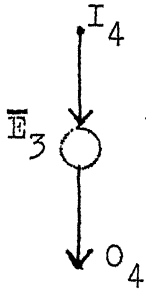

Step 2 Subgraph of gate $E_2$



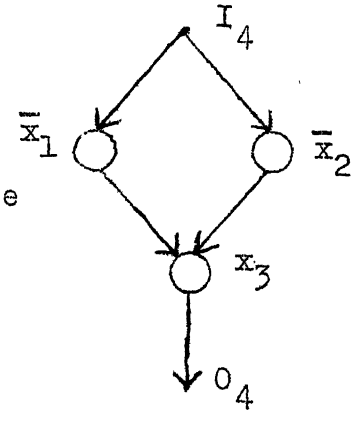

$E_1$ and $E_2$ replaced by their subgraphs




Step 3: Subgraph of gate $E_3$

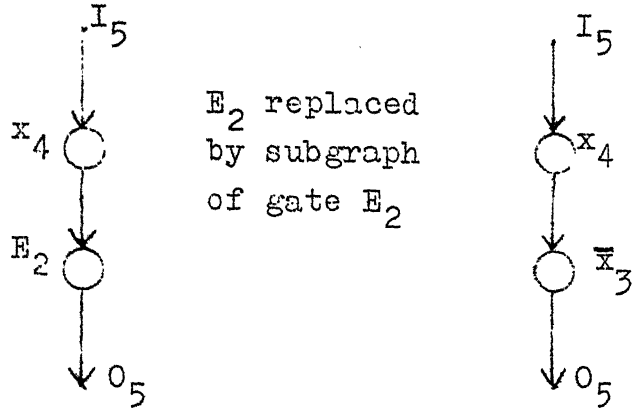



$\bar{E}_3$ replaced by complement subgraph of gate $E_3$

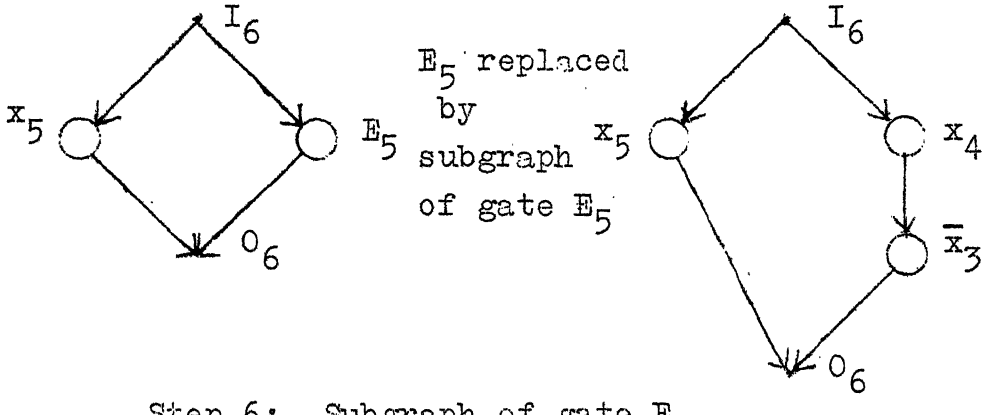



Step 4 : Subgraph of gate $E_4$

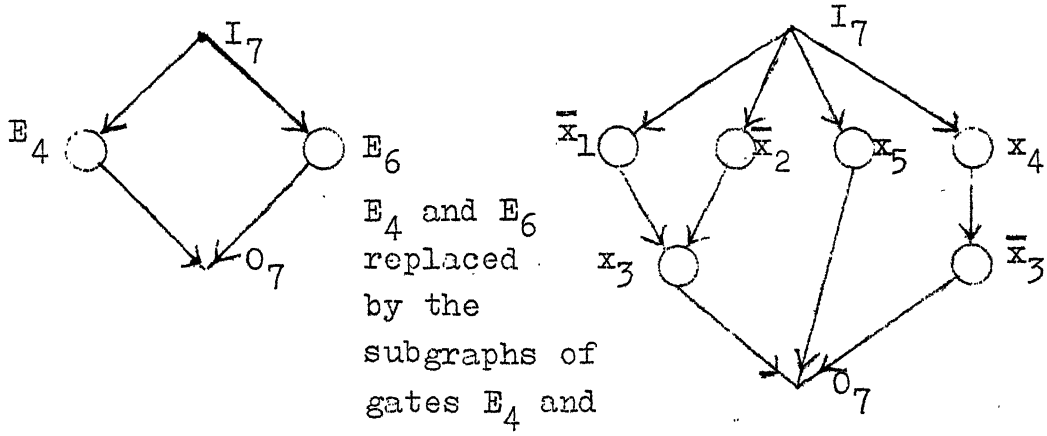
Figure 4.3




Step 5: Subgraph of gate $E_5$

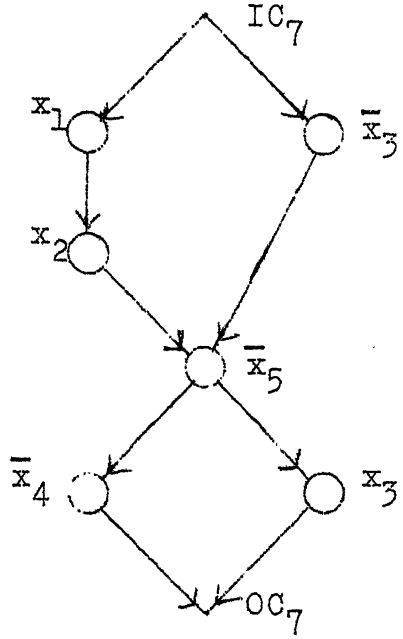



Step 6: Subgraph of gate $E_6$




Step 7 Subgraph of gate $E_7$-original graph of network

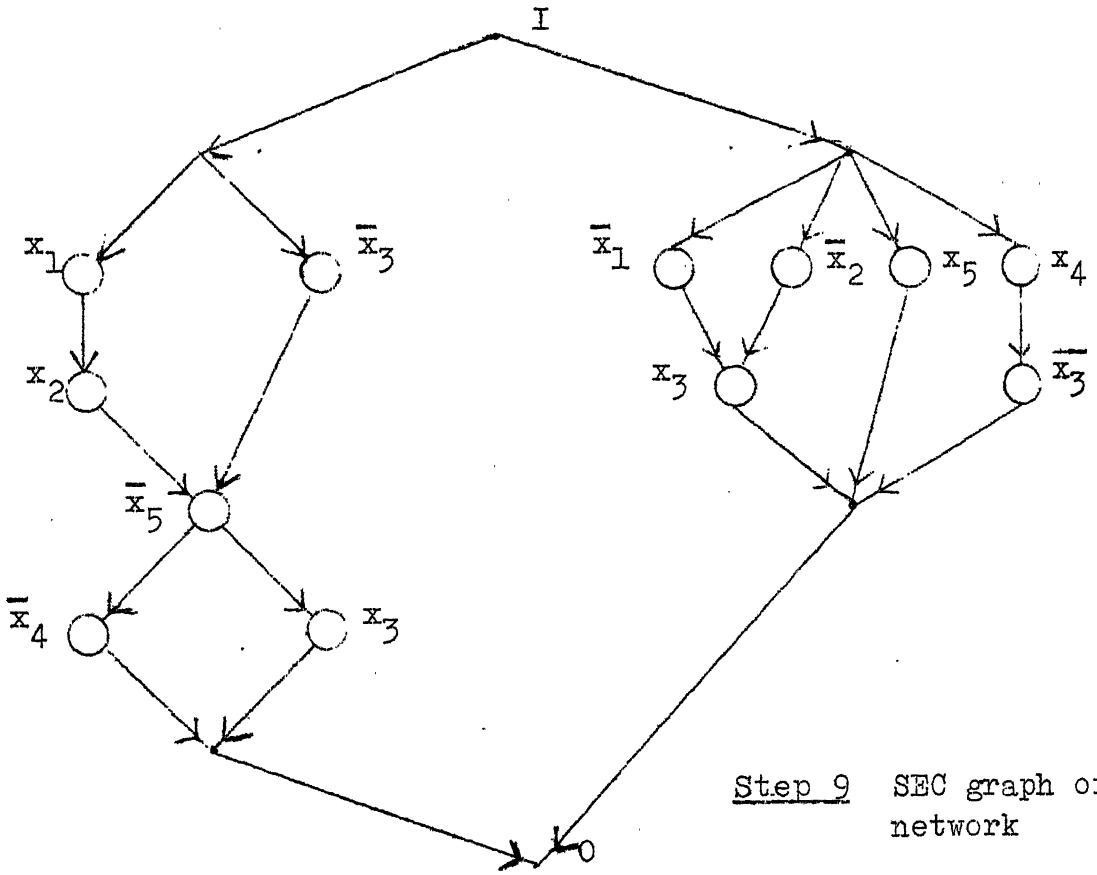
Figure 4.4 Subgraphs of gates $E_5$, $E_6$ and $E_7$.




$IC_7$- dummy input node of complement graph.

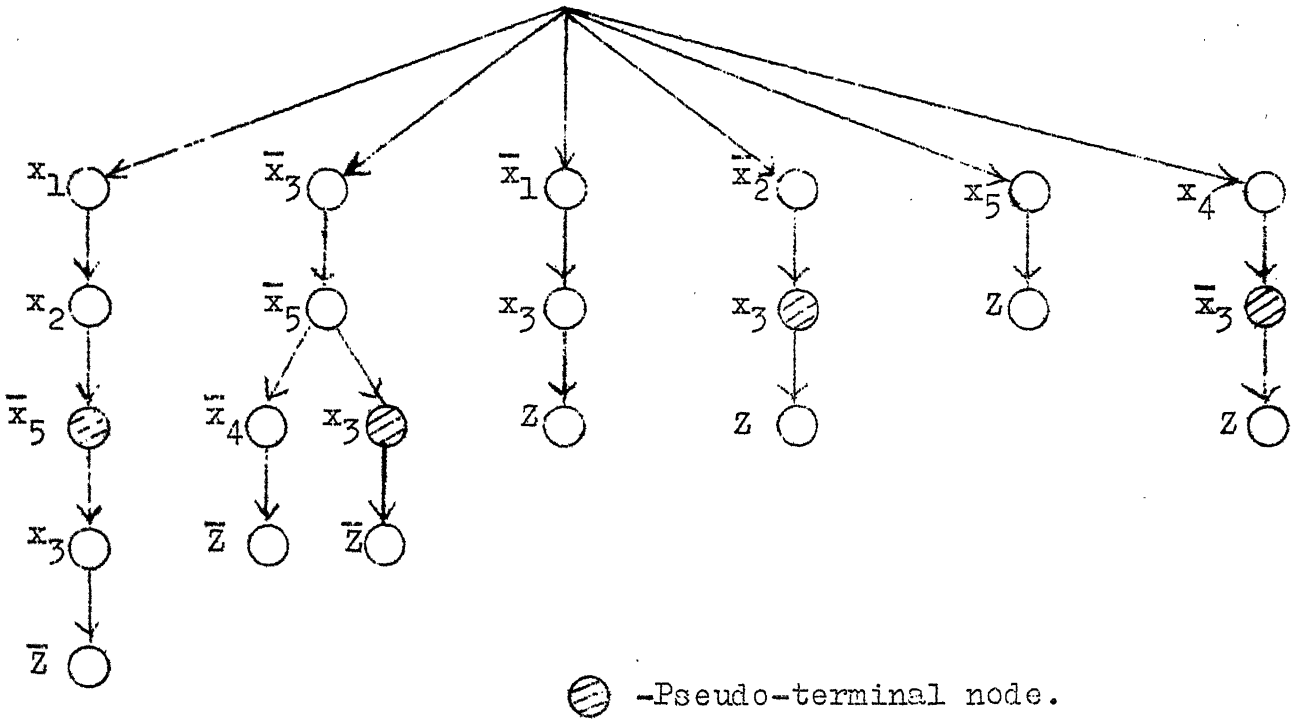
$OC_7$- dummy output node of complement graph.

Step 8 Complement graph of the network




Step 9 SEC graph of the network

Figure 4.5




Step 10 Paths generated in the SEC graph using PGM

Figure 4.6

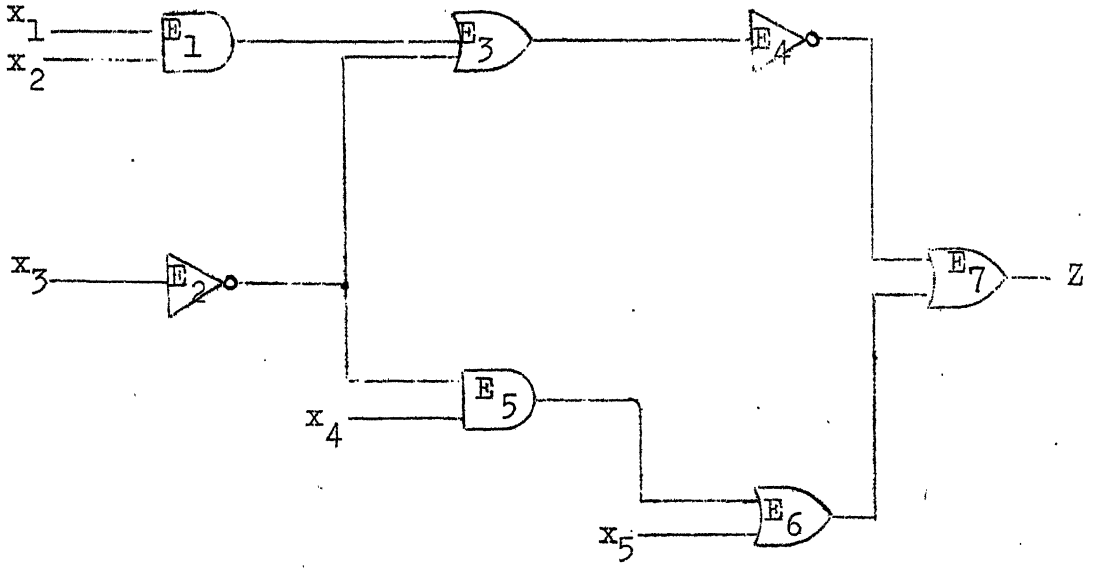



Figure 4.7 Example Network

The terminal node of paths in the original graph is marked Z. The terminal node of paths in the complement graph is marked $\bar{Z}$.

There is a conflict in the path $\bar{x}_3\bar{x}_5\bar{x}_3$ of the complement graph. This is caused by the reconvergent paths $E_2E_3E_4E_7$ and $E_2E_5E_6E_7$ in the network (shown in figure 4.7) having unequal inversions. This path is not considered further. The value assignment of the remaining paths is done and is given in table 4.3. The first 4 paths are in the original graph and the last 2 paths in the complement graph.

The DTS is shown as a diagnosing tree in figure 4.8. The total number of tests is 6 and the number of levels is 6. A total of 10 faults are detected and located by the 6 tests. This compares well with the path-sensitizing method, which would have required 10 iterations to generate tests for detecting and locating the 10 faults.

There are two sets of indistinguishable faults, ($x_1$ or $x_2$ s-a-0) and ($x_5$ or Z s-a-1).

In step 5 of the PGM (given in Sec.4.2), the choice of a path from a pseudo-terminal node to the output node is arbitrary. It should however be noted, that an arbitrary choic may create a sneak-path or a path with conflicts. This can be observed from figures 4.5(b) and 4.6 of the illustrative example. In figure 4.6 when the pseudo-terminal node $\bar{x}_5$ is

| Test-Path | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | Tests for |
|---|---|---|---|---|---|---|
| $t_1$ | 1 | 1 | 0 | 1 | 0 | $x_4$ and Z s-a-0 and $x_3$ s-a-1 |
| $t_2$ | 1 | 1 | 0 | 0 | 1 | $x_5$ and Z s-a-0 |
| $t_3$ | 1 | 0 | 1 | 0 | 0 | $x_3$ and Z s-a-0 and $x_2$ s-a-1 |
| $t_4$ | 0 | 1 | 1 | 0 | 0 | $x_3$ and Z s-a-0 and $x_1$ s-a-1 |
| $t_5$ | 1 | 1 | 1 | 1 | 0 | $x_1$, $x_2$ and $x_3$ s-a-0, and $x_5$ and Z s-a-1 |
| $t_6$ | 0 | 0 | 0 | 0 | 0 | $x_3$, $x_4$, $x_5$ and Z s-a-1 |

Table 4.3 Values assignment of the paths generated

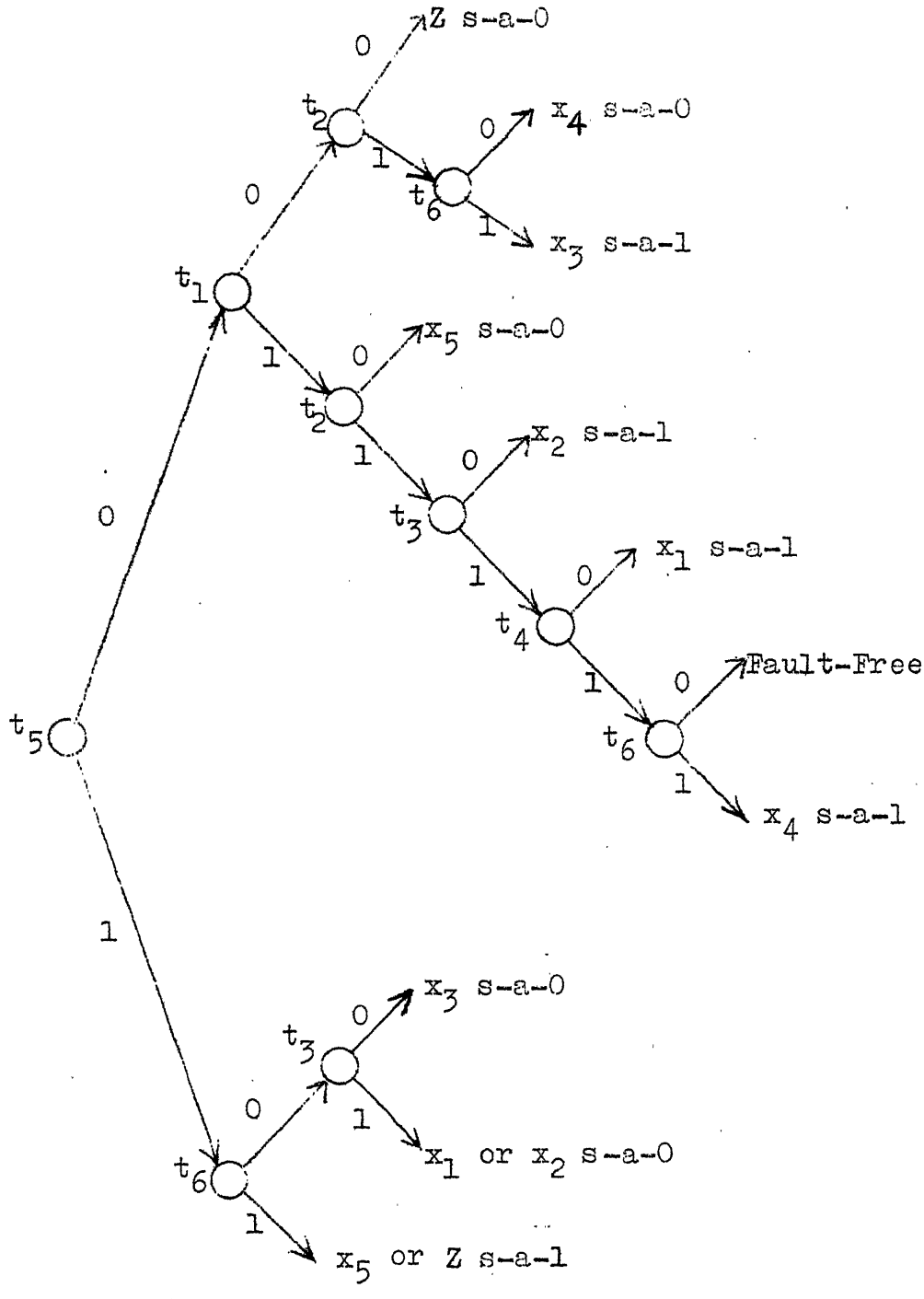



Figure 4.8 Diagnosing tree representation of the DTS

encountered, in the path $x_1x_2\bar{x}_5$ of the complement graph, there is a choice of reaching the output node via $x_3$ or $\bar{x}_4$. In our example we chose the path via $x_3$. Such a choice created no sneak path or conflicts. If path via node $\bar{x}_4$ had been taken, sneak-paths $\bar{x}_3\bar{x}_5\bar{x}_4$ or $x_1x_2\bar{x}_5x_3$ would have resulted.

Before ending this chapter, an error in the results of [10] should be pointed out. The path numbers 22-24, 26, 28-29, 31, 34-35, 38 and 40 have wrong values assigned, consequently, sneak-paths are created. The values could be reassigned to remove all sneak-paths. These errors are only in the paths of the complement graph. Therefore, they may have been caused by a discrepency in the program used to obtain the complement graph.

# CONCLUSION

An attempt was made in this thesis to study the uses of graph-theory in fault diagnosis. The conclusions of this study follow:

(1) Most of the problems in fault-location are caused by redundancy in the network or presence of reconvergent paths.

(2) The faults can be located to the smallest replaceable element if the facility of test-points or blocking-gates is provided.

(3) The PGM provides the most effective means to generate tests for large networks, having few redundancies and reconvergent paths.

## REFERENCES


1. ARMSTRONG, D.B. 'On finding a nearly minimal set of fault-detection tests for combinational logic nets'.
   IEEE Trans. El. Comp. EC-15, pp 66-73, 1966.

2. BENNETS, R.G., LEWIN., D.W. 'Fault diagnosis of Digital Systems'.
   COMPUTER JULY/AUGUST 1971.

3. CARROL, A.B., KATO, M., KOGA, Y. and NAEMURA, K. 'A method of diagnostic test generation.
   AFIPS Conf. Proc. SJCC VOL 34 pp 229-240, 1969.

4. CHANG, H.Y. 'An algorithm for selecting an optimum set of diagnostic tests'.
   IEEE Trans. El. Comp. EC-14, pp 706-711, 1965.

5. CHANG, H.Y., MANNING, E.G. and METZE, G. Fault Diagnosis in Digital Systems.
   WILEY-INTERSCIENCE, 1970.

6. DENT, J.J. ' Diagnostic Engineering Requirements'.
   AFIPS Conf. Proc. SJCC, pp 503-507 , 1968.

7. FRIEDMAN, A.D., and MENON, P.R. Fault detection in digital circuits.
   Prentice-Hall, Inc., EnglewoodCliffs, N.J. 1971.

8. HORNBUCKLE, G.D., and SPANN, R.M. 'Diagnosis of Single gate failures in combinational circuits'.
IEEE Trans. Comp., C-18, pp 216-220, 1969.

9. KAUTZ, W.H. 'Fault Testing and Diagnosis in Combinational Digital Circuits'.
IEEE Trans. El.Comp., EC-17, pp 352-368, 1968.

10. KOGA, Y., CHEN, C. and NAKMURA, K. 'A method of test generation for fault location in combinational logic'.
AFIPS Conf. Proc. FJCC, pp 69-78, 1970.

11. MAYEDA, W. and RAMAMOORTHY, C.V. 'Distinguishability Criteria in Oriented graphs and their application to Computer diagnosis'.
IEEE Trans. Circuit Theory, CT-16, pp 448-454, 1969.

12. McCLUSKEY, E.J., Jr. Introduction to the theory of switching circuits .
McGraw Hill, New York, 1965.

13. RAMAMOORTHY, C.V.'A structural theory of machine diagnostics
AFIPS Conf. Proc. SJCC, pp 743-756, 1967.

14. RAMAMOORTHY, C.V. 'Analysis of graphs by connectivity considerations'.
J. ACM, VOL. 13, pp 211-222, 1966.

15. RAMAMOORTHY, C.V. and MAYEDA, W. 'Computer diagnosis using the blocking-gate approach'.
IEEE Trans. Comp., C-20, pp 1294-1299, 1971.